НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
КАФЕДРА АВТОМАТИКИ ТА УПРАВЛІННЯ В ТЕХНІЧНИХ СИСТЕМАХ

«На правах рукопису»
УДК _004.4_____

«До захисту допущено»
Завідувач кафедри
_____ _О. І. Ролік___
(підпис) (ініціали, прізвище)
“___”_____20__ р.

Магістерська дисертація

зі спеціальності (спеціалізації) 121, інженерія програмного забезпечення (інженерія програмного забезпечення комп’ютерних систем)_____
(код і назва спеціальності)

на тему: Система моделювання завадостійких кодеків для телекомунікацій

Виконав (-ла): студент (-ка) __2__ курсу, групи ___ІТ-83мп_____
(шифр групи)

Кашапов Наіль Русланович _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник доцент кафедри АУТС, к.т.н, доцент Полторак В.П. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант _____ _____
(назва розділу) (науковий ступінь, вчене звання, , прізвище, ініціали) (підпис)

Рецензент в.о. зав. каф. СПіСКС, д.т.н, доцент Романкевич В. О. _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019 року

Національний технічний університет України
«Київський політехнічний інститут
імені Ігоря Сікорського»

Факультет (інститут) Інформатики та обчислювальної техніки
(повна назва)

Кафедра Автоматики та управління технічними системами
(повна назва)

Рівень вищої освіти – другий (магістерський) за освітньо-професійною програмою

Спеціальність (спеціалізація)_____
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О. І. Ролік
(підпис) (ініціали, прізвище)
«___»_____20__ р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Кашапову Наілю Руслановичу
(прізвище, ім'я, по батькові)

1. Тема дисертації Система моделювання завадостійких кодеків для телекомунікацій

науковий керівник дисертації Полторак В. П, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «___»_____ 20__ р. №_____

2. Строк подання студентом дисертації _____

3. Об'єкт дослідження робота завадостійких кодеків для телекомунікацій

4. Предмет дослідження (вихідні дані для магістерської дисертації за освітньо-професійною програмою) покращення процесу моделювання завадостійких кодеків та інструментів для їх вивчення

5. Перелік завдань, які потрібно розробити ознайомитися з принципами завадостійкого кодування, проаналізувати існуючі аналогічні рішення, визначити необхідні вимоги, розглянути необхідні архітектурні та

технологічні рішення, реалізувати роботи основних кодів, розробити програмну систему для моделювання завадостійких кодів

6. Орієнтовний перелік ілюстративного (графічного) матеріалу Діаграма прецедентів системи, ER-діаграма, діаграма компонентів, діаграма пакетів, діаграма послідовності процесу авторизації

7. Орієнтовний перелік публікацій _____

8. Консультанти розділів дисертації[1*]

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

9. Дата видачі завдання 24 жовтня 2018

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів магістерської дисертації	Примітка
1	Порівняльний аналіз існуючих рішень	02.09.19 - 06.09.19	
2	Визначення основних вимог до системи	09.09.19 - 13.09.19	
3	Розроблення сценаріїв використання системи	16.09.19 - 20.09.19	
4	Вибір технологій для розробки	23.09.19 - 27.09.19	
5	Розроблення структурної схеми	30.09.19 - 04.10.19	
6	Розроблення ER-діаграми	07.10.19 - 11.10.19	
7	Реалізація бізнес-логіки	14.10.19 - 18.10.19	
8	Розроблення інтерфейсу користувача	21.10.19 - 25.10.19	
9	Розроблення стартап-проекту	28.10.19 - 02.11.19	
10	Оформлення текстового матеріалу	04.11.19 - 25.11.19	
11	Оформлення графічного матеріалу	26.11.19 - 02.12.19	
12	Подача дисертації на перевірку	03.12.19 - 18.12.19	

Студент _____ Кашапов Н. Р.
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ Полторак В. П.
(підпис) (ініціали, прізвище)

[1*] Консультантом не може бути зазначено наукового керівника

РЕФЕРАТ

Магістерська дисертація містить 112 сторінок пояснювальної записки, 16 рисунків, 40 таблиць, 8 креслеників та 29 посилань на використані літературні джерела.

Актуальність даної роботи полягає в тому, що на даний момент існує обмежена кількість сучасних інструментів для вивчення моделювання завадостійкого кодування, в той час як воно все ще поширене в телекомунікаційних системах. Деякі існуючі рішення надають багатофункціональні, дорогі програмні пакети, орієнтовані на комплексні обчислення та моделювання. Інші не відповідають стандартам якості та мають значно обмежений функціонал.

Мета дисертації – розробка нових інструментів для покращення процесу вивчення роботи завадостійкого кодування.

Об'єктом дослідження дисертації є робота завадостійких кодів для телекомунікацій. Предметом дослідження є покращення процесу моделювання завадостійких кодеків та інструментів для їх вивчення.

При вирішенні поставлених завдань застосовувалися загальні методи наукового пізнання такі, як спостереження, вимірювання та порівняння, а також методи аналізу і синтезу. За їх допомогою було досліджено існуючі аналоги, визначено основні шляхи для покращення.

Результати роботи можуть бути використаними для доповнення загальної розподіленої системи, призначеної для навчальної діяльності. Дана система може використовуватися і в якості модуля більшого комплексу, так і працювати незалежно. Проведені у дисертації дослідження та розроблені рішення можуть бути використані не лише для дослідницьких та навчальних процесів, а і використовуватися на практиці.

Ключові слова: архітектура, веб-застосунок, завадостійке кодування, кодек, моделювання, односторінковий додаток, Vue компонент.

SUMMARY

The master's dissertation contains 112 sheets of explanatory note, 16 pictures, 40 tables, 8 drawings and 22 bibliographic references on used literary sources.

The relevance of this work lies in the fact, that to this day, amount of modern tools for learning redundant coding modeling is limited, while its still broadly used in telecommunications systems. Some existing solutions provide multifunctional, expensive software packages focused on complex computing and modeling. Others do not meet quality standards and have significantly limited functionality.

The purpose of the dissertation is to develop new tools to improve the learning process of noise immunity coding.

The object of research is the work of redundant codes for telecommunications. The subject of the study is to improve the process of redundant codecs modeling and tools for studying them.

Common methods of scientific knowledge, such as observation, measurement and comparison, as well as methods of analysis and synthesis, were used to solve these tasks. With their help, the existing solutions were investigated, and the main ways for improvement were identified.

The results of the work can be used to supplement the completed distributed system for educational activities. This system can be used as a module of a larger complex, and can also work independently. The research carried out in the dissertation and the solutions developed can be used not only for research and educational processes, but also for practical use.

Keywords: architecture, web application, redundant coding, codec, modeling, one-page application, Vue component.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

AJAX – Asynchronous JavaScript and XML

API – Application Programming Interface

CDN – Content Delivery Network

CSS – Cascade Style Sheets

DOM – Document Object Model

HTML – Hypertext Markup Language

HTTP – Hypertext Transfer Protocol

HTTPS – Hypertext Transfer Protocol Secure

JSON – JavaScript Object Notation

JSX – JavaScript XML

MVC – Model View Controller

MVVM – Model View ViewModel

SPA – Single Page Application

SQL – Structured Query Language

XML – eXtensible Markup Language

URL – Uniform Resource Locator

ВСТУП

На сьогоднішній день телекомунікації є важливим засобом для обміну даними. З початку 20-го століття почалася стрімка революція у галузях електричних комунікацій. В той час як були винайдені телефони, телеграфи, радіозв'язок і телебачення, методи дистанційного обміну інформації набували все більшого поширення. Сучасні засоби дозволяють отримати доступ до інформації, що зберігається на сервері за тисячі кілометрів через мережу Інтернет.

Але з розвитком і поширенням засобів комунікацій, більш гострішим постає питання захисту та надійності передачі даних. На сьогоднішній день через Інтернет поширюється величезній об'єм інформації. Для безпечного відправлення та отримання повідомлення використовуються різноманітні шляхи, один з яких кодування вихідного сигналу. Кодування – це процес перетворення повідомлення на впорядкований набір символів, елементів, знаків. При кодуванні кожному повідомленню ставиться у відповідність зумовлена кодова комбінація [1].

Однак жодна передача даних не застрахована від помилки – радіоперешкоди, ненадійність каналів зв'язку, несправність обладнання. Для виявлення і виправлення помилок при передачі повідомлення використовуються завадостійкі коди.

Завадостійке кодування і декодування широко поширене у ненадійних каналах зв'язку, наприклад телефонних лініях. Такі канали просто не здатні знову відправити інформацію у випадку виявлені помилок. Якщо у повідомленні виникла помилка, оригінальна інформація більше не доступна. Також завадостійкі кодеки широко поширені у системах збереження інформації – оптичних і магнітних.

Таким чином, завадостійке кодування дійсно має свої способи застосування, проте, на даний момент способи його вивчення та дослідження не відповідають сучасним вимогам. На даний момент, аналоги не досягають відповідного рівня розвитку технологій, хоча існують можливості для їх покращення і оновлення. Для деталізованого вивчення процесу виявлення і виправлення помилок завадостійкими кодеками було вирішено розробити систему для моделювання їх роботи.

Мета дисертації – розробка нових інструментів для покращення процесу вивчення

роботи завадостійкого кодування. Для досягнення поставленої мети були визначені наступні завдання:

- ознайомитися з основними принципами завадостійкого кодування і декодування;
- проаналізувати існуючі аналогічні рішення та визначити особливості, які впливають на якість процесу моделювання;
- визначити необхідні вимоги до системи, що дозволить виділити головні шляхи для розробки;
- розглянути найбільш відповідні архітектурні та технологічні рішення для даної системи, та вибрати такі, які краще підходять для розв'язання поставлені завдання;
- реалізувати кодування і декодування основних завадостійких кодів;
- використовуючи отримані дані, розробити програмне забезпечення для системи моделювання завадостійких кодеків для телекомунікацій, що дозволить покращити навчальні процеси вивчення і дослідження, зробити їх зручнішими і легшими для використання.

Об'єктом дослідження дисертації є робота завадостійких кодів для телекомунікацій. Предметом дослідження є покращення процесу моделювання завадостійких кодеків та інструментів для їх вивчення.

При вирішенні поставлених завдань застосовувалися загальні методи наукового пізнання такі, як спостереження, вимірювання та порівняння, а також методи аналізу і синтезу. За їх допомогою було досліджено існуючі аналоги, визначено основні шляхи для покращення.

Результати роботи можуть бути використаними для доповнення загальної розподіленої системи, призначеної для навчальної діяльності. Дана система може використовуватися і в якості модуля більшого комплексу, так і працювати незалежно. Проведені у дисертації дослідження та розроблені рішення можуть бути використані не лише для дослідницьких та навчальних процесів, а і використовуватися на практиці.

1 ОГЛЯД ТА АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

Для формування вимог до системи, що розробляється необхідно проаналізувати зразки подібних систем. Детальний аналіз допоможе сформувати перелік необхідного функціоналу, виявити переваги та недоліки подібних систем.

На даний момент існує не так багато систем моделювання які дозволяють вивчати роботу завадостійких кодеків. Існують декілька багатофункціональних, дорогих програмних пакетів, з широким функціональним спектром, за допомогою яких можна виконати обчислення в полях Галуа для кодоперетворення циклічних кодів. Ці програми універсальні, тому вони складні в експлуатації, оскільки вирішують велику кількість різнотипових задач. Яскравими прикладами таких пакетів є MatLab та Volfram Mathematica [2]. Тому, було розглянуто більш прості аналогічні системи, які орієнтовані на моделювання.

1.1 Система моделювання кодеків XTest+

Система XTest+ призначена для моделювання роботи кодеків в навчальних цілях. XTest+ виконаний в якості настільного додатку на мові програмування Java. Додаток скомпільований в виконуваний .jar файл, що дозволяє запускати програму на більшості операційних систем, за умови встановленого середовища виконання Java. Проте, система використовує велику кількість сторонніх бібліотек, тому потребує поширення у вигляді архіву. Його вага не є критичною, але такий спосіб не зручний для багатьох користувачів.

Система охоплює широкий спектр завадостійких кодів такі як систематичні, циклічні та недвійкові, а також декілька інших. Основне призначення системи полягає в наданні користувачам теоретичних відомостей про наявні кодеки, моделювання їх роботи та перевірка знань і практичних навичок користувачів за допомогою тестувань.

Інтерфейс настільного додатку є мінімалістичним і простим, проте має незначні недоліки. При запуску програми, користувач буде представлений головному меню,

яке містить список наявних кодеків, розбитих на відповідні категорії, та чотири кнопки, які дозволяють переглянути теоретичні відомості, перейти до виконання практичних завдань, пройти тестування, або згенерувати звіт. Для того, щоби скористатися функціоналом додатку, користувачу необхідно вибрати потрібний кодек зі списку, в протилежному випадку, кнопки для роботи з кодеком будуть просто неактивними. Також, користувач не може змінити розмір вікна додатку та збільшити шрифт. В цілому, дизайн користувацького інтерфейсу не відповідає сучасним ергономічним вимогам [3]. Головне меню додатку зображено на рисунку 1.1.

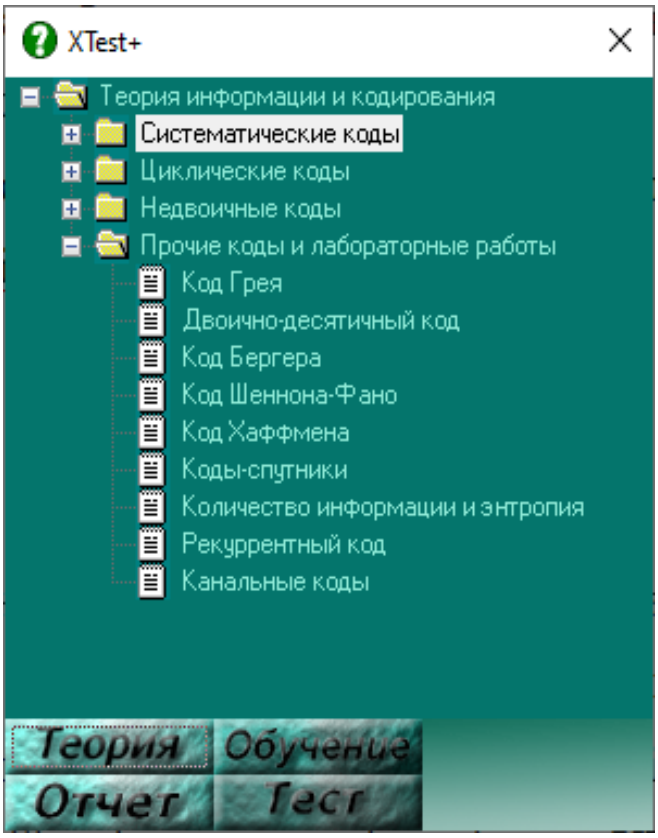

Рисунок 1.1 – Головне меню додатку XTest+

В розділі теорії користувач може переглянути необхідні для проходження тестування теоретичні відомості вибраного кодеку. Більшість кодів мають непогано структуровану теорію, а також декілька прикладів кодування і декодування. Проте, деякі коди не мають прикладів, а викладена теорія обмежується загальним описом, що погіршує розуміння та вивчення матеріалу.

Теоретичні матеріали відкриваються в окремому вікні, яке на відміну від головного меню, можна збільшувати при необхідності, та залишати відкритим під час виконання тестування, що доволі зручно. Проте, усі теоретичні матеріали викладені російською мовою, що може перешкодити вивченню теорії для деяких користувачів. Наявність українського перекладу, або можливість змінювати мову додатку була б корисною функцією для гарного засвоєнню матеріалу усіма користувачами. Приклад вікна з теоретичними відомостями зображено на рисунку 1.2.

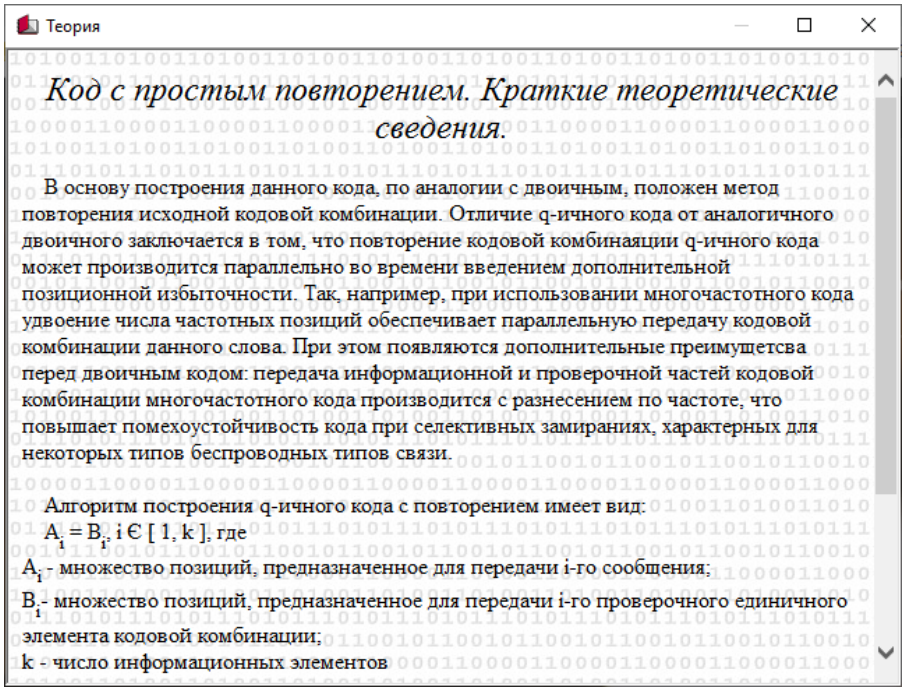

Рисунок 1.2 – Вікно з теоретичними відомостями коду

Після ознайомлення з теоретичними відомостями про певний код, користувач може виконати завдання кодування або декодування, для кращого розуміння практичної роботи кодеків. Більшість практичних завдань представлені у вигляді невеликого вікна, з випадково згенерованою умовою. Після введення відповіді, користувач отримає повідомлення, яке сповістить його про правильність виконання.

Одним із найбільших недоліків системи XTest+ є відсутність можливості змінювати розмір шрифту. Зі збільшенням довжини повідомлення з умови, перерахувати кількість символів стає все більш проблематичним, особливо, якщо

повідомлення представлено двійковим кодом.

Також, генерування умови випадково, безперечно має свої переваги, оскільки жоден користувач не отримає одного і того ж завдання двічі. Проте, в деяких кодеках, які містять декілька підвидів, відсутність функції інтерфейсу вибрати певний підтип, змушує користувача закривати і відкривати вікно з практичним завданням знову, поки він не отримає умови з потрібним підвидом коду. Можливість змінювати певні параметри генерування умови для тренувальних вправ була би доцільною функцією системи моделювання. Приклад практичного завдання зображено на рисунку 1.3.

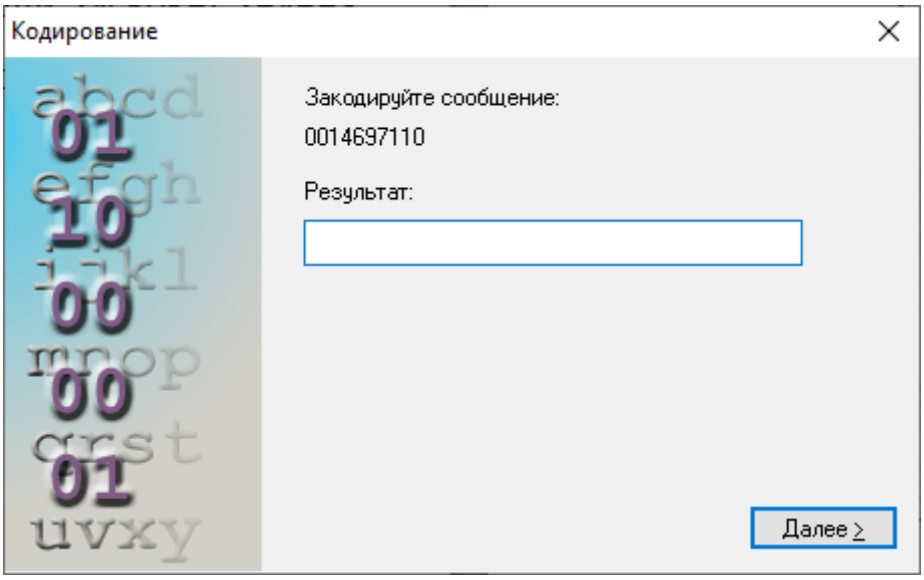

Рисунок 1.3 – Вікно з практичним завданням

Після набуття практичних навичок, для випробування засвоєних знань, користувач повинен пройти тестування. Тестування представляє собою набір з десяти завдань, п'ять на кодування, п'ять на декодування, ідентичних тренувальним. На відміну від тренувальних завдань, користувач не повідомляється про правильність чи неправильність виконаного завдання.

Після виконання тестування, користувач отримає оцінку, засновану на кількості правильно виконаних завдань. Проте система не дозволяє перевірити, які саме відповіді були неправильними. Результати тестування зберігаються під іменем локального користувача комп'ютера, на якому запущений настільний додаток.

Користувач може переглянути свої результати в розділі звіту, де зберігаються оцінки за пройдені тестування. Система надає можливість роздрукувати даний звіт, в іншому випадку, користувач повинен зберегти скріншот звіту, оскільки додаток XTest+ не здатний зберігати інформацію про користувача за межами даної сесії. Вікно звіту зображено на рисунку 1.4.

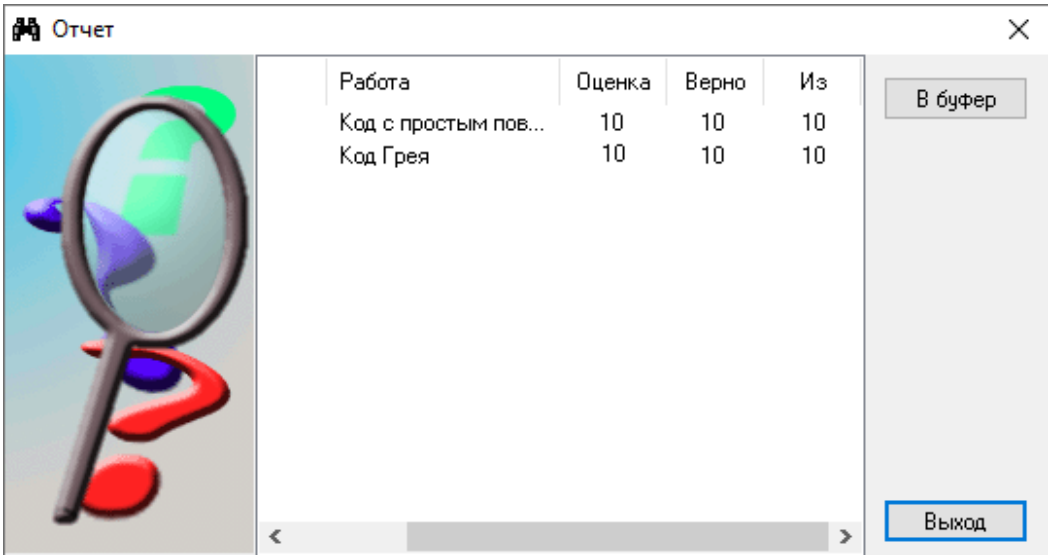

Рисунок 1.4 – Звіт виконаних тестувань

Після закриття додатку, інформація про пройдені тести буде знищена і недоступна для відтворення. Це є серйозним недоліком для настільного додатку який призначений для особистого користування. Система XTest+ не здатна перевірити, чи було справді пройдено конкретне тестування, і надати отриману оцінку за нього. Таким чином, настільний додаток є вразливим для програмного забезпечення, яке може маніпулювати комп'ютерною пам'яттю, на зразок CheatEngine. Використовуючи таке обладнання, зловмисні користувачі можуть підробити звіт, згенерований системою, підставивши потрібні значення.

В цілому, програмний додаток працює без затримок, нові вікна відкриваються швидко, обчислення не займають багато часу. Існує невелика затримка при завершенні тестування, під час якої відбувається збереження результатів, але вона є незначною.

Після детального огляду настільного додатку, з переваг системи XTest+ можна

виділити:

- кросплатформеність;
- невелика вага настільного застосунку;
- широкий набір реалізованих кодеків;
- доступ до теоретичних даних та тестування;

З недоліків можна виділити:

- застарілий і незручний інтерфейс;
- складність поширення додатку;
- збереження даних користувача лише в межах однієї сесії;
- урізані можливості для практичних завдань, умови генеруються випадково;
- вразливість до програмного забезпечення яке може маніпулювати комп'ютерною пам'яттю;
- неможливість отримання інформації про виконані тестування;
- відсутність українського перекладу.

1.2 Система моделювання і обробки сигналів SignalJ

Система SignalJ орієнтована на використання студентами для виконання практичних завдань навчальних курсів. Система надає широкий та багатий функціонал для дослідження різних сигналів, виконання математичних та технічних операцій на них, та детального вивчення результату. Основним завдання системи є створення сигналів за допомогою функцій, формули або вручну та їх подальша обробка.

Система виконана в якості настільного додатку за допомогою мови програмування Java, і скомпільована у виконуваний .jar файл, що дозволяє запускати додаток на більшості операційних систем. SignalJ також залежний від зовнішніх бібліотек, проте, на відміну від попереднього аналогу, вони також скомпільовані в .jar архіви. Вага усього архіву невелика, але поширення додатку не є зручним.

Інтерфейс програми досить складний, проте інтуїтивний. На верхній панелі представлені основні функції програми – створити сигнал, його редагування або

обробка, відобразити його у вигляді графіку або таблиці, допоміжні функції та налаштування. Нижче головної панелі існує допоміжна з іконками які представляють найбільш популярні операції – зберегти, скопіювати, вставити, закрити вікно тощо.

Для моделювання сигналів система SignalJ надає широкий вибір для створення та налаштування джерел сигналу. Для створення можна використовувати не тільки свої формули, які дуже зручно вводити в даному інтерфейсі, але також застосовувати вже готові стандартні типи сигналів, не витрачаючи час на їх опис функцією. Система навіть дозволяє створювати шуми з різними параметрами. Меню створення стандартного сигналу зображено на рисунку 1.5.

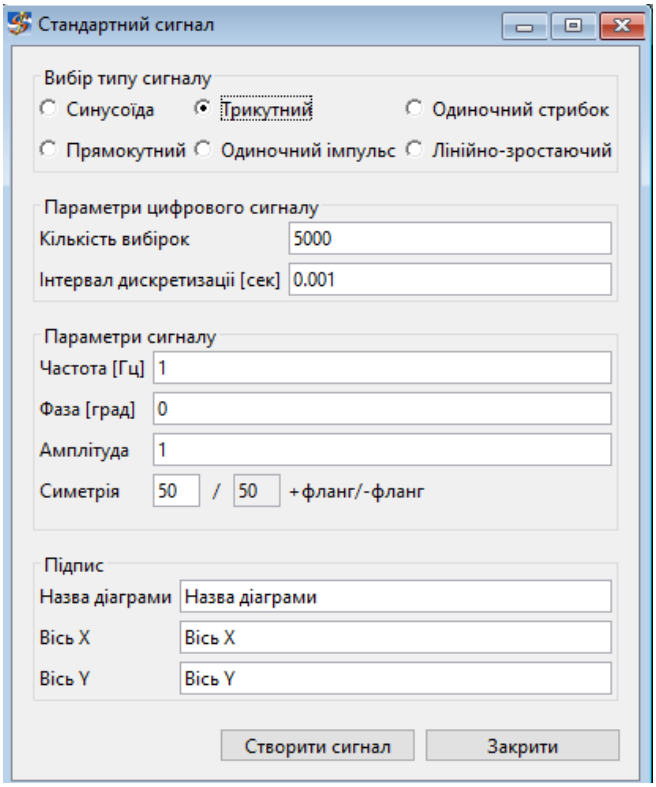

Рисунок 1.5 – Меню створення стандартного сигналу

Після створення сигналу, користувач може вибрати вид його представлення. Система дозволяє зобразити сигнал у вигляді таблиці, лінійного графіку, або стовпчикової діаграми. Проте деякі сигнали, зокрема шуми, не можуть бути представлені у певному вигляді, або втратять сенс у вибраному представлені.

Користувач також має можливість налаштувати представлення сигналу. У випадку графіка, система надає можливість змінити тип діаграми, налаштувати

вигляд осей, тип лінії, інтервал та координатну сітку. Зручний інтерфейс дозволяє тут же застосувати зміни і перемалювати діаграму. Швидкість оновлення діаграми залежить від складності формули яка її задає, і параметрів координатних осей. Проте навіть в складних випадках, оновлення діаграми не займає більше кількох секунд, а продуктивність додатку починає сповільнюватися при кількості відкритих вікон графіків більше за 15. Представлення сигналу у вигляді лінійного графіку та меню його налаштування зображено на рисунку 1.6.

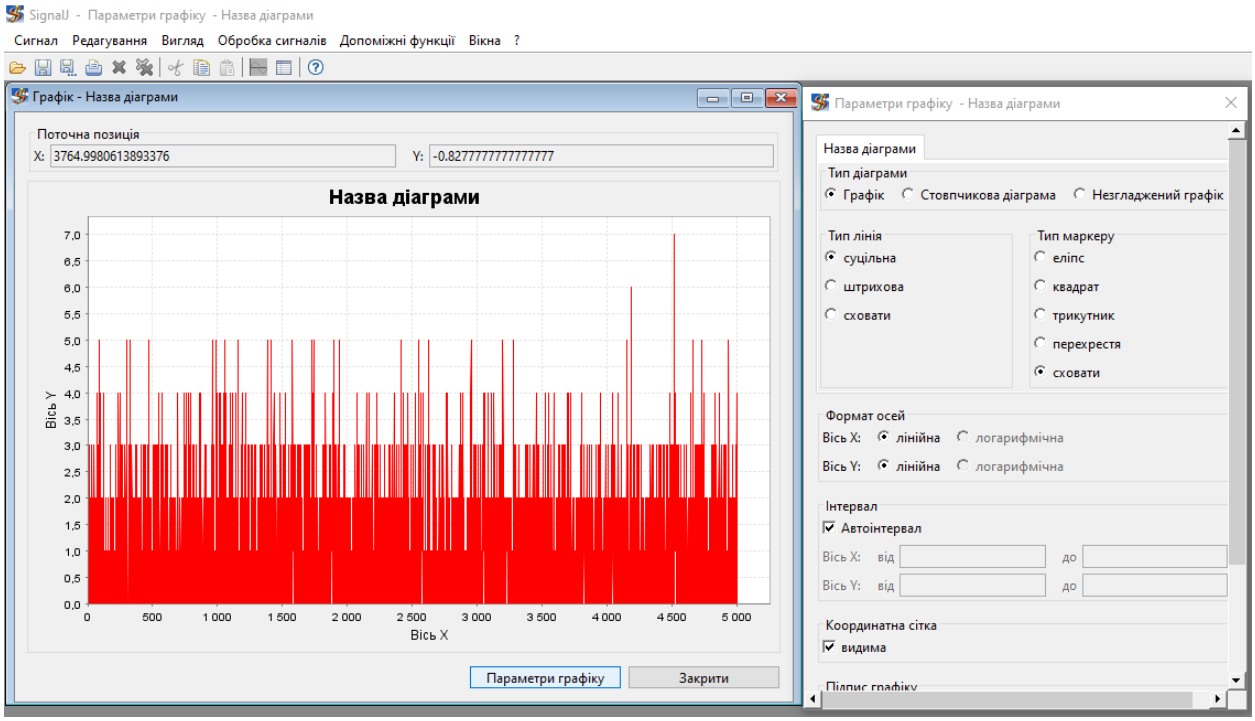

Рисунок 1.6 – Графік сигналу та меню його налаштування

Створення різноманітних сигналів є лише частиною функціоналу системи. SignalJ також дозволяє користувачу виконувати широкий спектр операцій на створених сигналах. Обробка сигналів доступна під однойменною вкладкою, та містить інтерполяцію, апроксимацію та багато інших. Після вибору необхідної операції та його налаштування, в окремому вікні програми буде відображений графік, який зображує результат здійсненої операції.

Для кожного виду обробки в системі представлений повний інтерфейс для його налаштування. Наприклад для ряду Фур'є користувач може вибрати кількість гармонік, тривалість періоду, суму помилок, а також переглянути таблицю

коефіцієнтів Фур'є, і перерахувати ряд при бажанні. На рисунку 1.7 зображений графік апроксимації за рядом Фур'є синусоїдного сигналу.

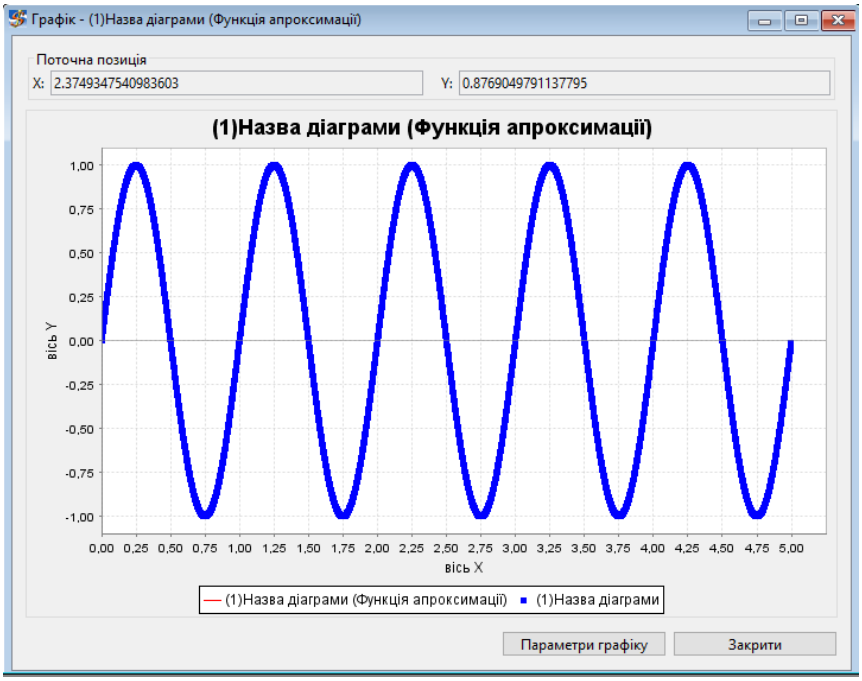

Рисунок 1.7 – Графік апроксимації за рядом Фур'є синусоїдного сигналу

Однією за найбільших переваг системи моделювання SignalJ є можливість збереження будь-якого сигналу. Дані, на основі яких будується графік, зберігаються у файл з розширенням .xml. Під час завантаження файлу у програму, на основі збережених даних відновлюється графік сигналу.

Це дозволяє зберігати процес виконання практичних завдань і продовжувати його пізніше, а також зменшує можливість постраждати при помилковому збою програми. Для користувача це можливість зберегти свою роботу поетапно, для підтвердження виконаного практичного завдання. Проте, більшість передбачених навчальним курсом завдань передбачає використання великої кількості сигналів і графіків, ручне збереження яких і їх подальше завантаження у систему, може бути нудною і незручною операцією. При відкритті програми, відсутня функція автоматично завантажити потрібні сигнали. Можливість зберігати стан системи разом з усіма налаштуваннями сигналів і їх графіками в окремий файл була би більш доцільнішою.

Після детального огляду системи SignalJ можна виділити наступні переваги:

- широкий функціонал системи;
- зручний і детальний користувацький інтерфейс;
- можливість детального налаштування процесу моделювання сигналів;
- можливість збереження результатів роботи;
- висока швидкість виконання обчислень;
- можливість вибору видів представлень сигналу.

З недоліків:

- відсутність теоретичних відомостей;
- незручність збереження великої кількості сигналів і їх подальшого завантаження в програму;
- вразливість до програмного забезпечення яке може маніпулювати комп'ютерною пам'яттю;
- неточний переклад деяких елементів інтерфейсу.

1.3 Порівняльний аналіз аналогічних продуктів

Аналіз вище переглянутих аналогів показав їх позитивні і негативні якості. Використовуючи ці фактори, була складена таблиця порівняння аналогів та розроблюваної системи (таблиця 1.1).

Таблиця 1.1 – Таблиця порівняння аналогів

Критерій оцінки	XTest+	SignalJ	Розроблювана система
Наявність теоретичних відомостей	+	–	+
Можливість налаштування процесу моделювання	–	+	+

Продовження таблиці 1.1.

Критерій оцінки	XTest+	SignalJ	Розроблювана система
Наявність тестування знань користувача	+	–	+
Збереження результатів роботи	+	+	+
Наявність українського перекладу	–	+	+
Зручний інтерфейс	–	+	+
Широкий функціонал системи	–	+	+
Гнучка архітектура для розширення системи	+	–	+
Кросс-платформеність системи	+	+	+
Можливість швидкого портування застосунку	–	–	+
Доступу до системи офлайн	+	+	–

1.4 Висновки до розділу

Під час огляду та аналізу аналогів, буди розглянуті існуючи продукти, їх переваги і недоліки. Система моделювання кодеків XTest+ хоч і надає широкий вибір кодів для моделювання та інструменти для їх тестування, має не ергономічний інтерфейс, незручна для використання, не має можливості зберігати дані користувача, та не містить ряд важливих функцій необхідних для подібних систем. Було встановлено, що система XTest+ не відповідає сучасним вимогам, а її подальша розробка і підтримка не є цілеспрямованою.

Система моделювання і обробки сигналів SignalJ, незважаючи на недоліки, пов'язані з настільним додатком, має зручний і детальний інструментарій для роботи з цифровими сигналами. Переваги даної системи переважують її недоліки і SignalJ може слугувати стійким прикладом для розроблюваної системи для моделювання завадостійких кодеків.

Обидві системи виконані в якості настільних портативних додатків, що не потребують встановлення на локальну машину користувача. Це, звісно, зменшує час на встановлення, проте створює ряд вразливостей для програмного забезпечення, яке може отримати доступ до пам'яті комп'ютера.

Таким чином, була встановлена доцільність розробки нової, більш досконалої системи. Зважаючи на аналіз аналогів, були встановлені їх основні проблеми, які повинна вирішувати нова система: зберігати дані користувача та результати його роботи із застосунком та забезпечувати їх безпеку та недоступність, мати зручний інтерфейс, який дозволить користувачам вносити зміни в процес моделювання кодеків, мати гнучку архітектуру з можливістю доповнення готової системи новими модулями. Також, система повинна бути кросплатформенною, мати український переклад, таким чином бути доступними для всіх користувачів. Ці критерії є найголовнішими, і будуть визначати пріоритети під час розробки. Іншими, менш важливими недоліками, можна знехтувати.

2 ФОРМУВАННЯ ВИМОГ ДО СИСТЕМИ

Формування вимог до розроблюваної системи є важливим етапом проектування програмного забезпечення. Вимоги визначають властивості, якості та функції системи. Чітко встановлені вимоги допоможуть розбити процес розробки на логічні етапи, встановити пріоритет реалізації функцій програмного забезпечення.

Вимоги поділяються на функціональні та нефункціональні. Спершу, встановимо функціональні вимоги, а на їх основі визначимо нефункціональні.

2.1 Формування функціональних вимог

Функціональні вимоги описують поведінку програмного забезпечення, його внутрішній функціонал. Вони визначають, що саме повинна виконувати розроблювана система [4]. Як було встановлено в попередньому розділу, однією з проблем, яка потребує вирішення, є збереження користувача, його даних та результатів роботи в системі. Створення механізму авторизації через облікові записи дозволить кожному користувачу не тільки отримати доступ до своїх даних, а і допоможе з безпекою системи, що також є важливою проблемою.

Також, дана система повинна надавати зручний інтерфейс, та широкий функціонал для роботи з кодеками. Тому для були визначені наступні функціональні вимоги:

- забезпечити реєстрацію нового користувача з використанням його електронної адреси;
- при створенні нового облікового запису зберегти такі персональні дані користувача як ім'я, прізвище і групу;
- після створення нового облікового запису, відправити на електрону адресу користувача повідомлення з підтвердженням;
- забезпечити авторизацію існуючого користувача;
- забезпечити можливість вийти з авторизованого облікового запису;
- після вдалої авторизації надати користувачу доступ до функціоналу системи;

- забезпечити наявність українського перекладу;
- надати до перегляду список наявних в системі кодів;
- надати можливість вибрати потрібний код зі списку і перейти на його сторінку;
- надати можливість переглянути відповідні теоретичні відомості та приклади роботи кодеку;
- надати можливість виконати відповідні тренувальні завдання кодування і декодування;
- надати можливість налаштувати параметри генерування умови для практичного завдання;
- надати можливість переглянути кількість виконаних тренувальних вправ для вибраного коду;
- надати можливість пройти тестування з вибраного коду;
- відраховувати час, затрачений на проходження тестування;
- оцінити пройдене користувачем тестування;
- надати можливість переглянути набрану оцінку за тестування;
- надати можливість переглянути кількість спроб з вибраного тестування;
- надати можливість перепройти вибране тестування;
- надати доступ про інформацію про дану версію системи;
- надати доступ до перегляду сторінки з довідкою.

2.2 Формування нефункціональних вимог

На відміну від функціональних, нефункціональні вимоги визначають як саме програмне забезпечення повинно виконувати свою роботу і які властивості і характеристики воно повинно мати [4, 5]. Стосовно розроблюваної системи, нефункціональні вимоги будуть описувати якісні показники програмного забезпечення. Для даної системи були визначені наступні вимоги:

- повідомлення про успішне створення облікового запису повинно прийти на електрону пошту за час, не більший ніж п'ять хвилин після завершення реєстрації;

- при успішній авторизації користувач має перейти на головне меню системи за час, не більший ніж одну секунду;

- при реєстрації виконати перевірку на правильність введеної електронної адреси;

- після виходу користувача зі свого облікового запису, система повинна перейти на вікно авторизації за час, не більший ніж одну секунду.

- функціонал системи повинен бути доступним користувачам будь-яких операційних систем;

- забезпечити перехід з однієї сторінки системи на іншу за час, не менший ніж дві секунди;

- завантаження і коректне представлення даних користувача за час, не менший ніж дві секунди;

- умови практичних завдань повинні генеруватися довжиною, встановленою користувачем, і за час, не більший ніж одну секунду;

- похибка таймера, який відміряє час затрачений на тестування, повинна бути не більша за три секунди;

- система повинна перевірити виконане користувачем тестування і надати відповідну оцінку за час, не менший ніж дві секунди.

2.3 Висновки до розділу

В даному розділі, на основі попереднього аналізу переваг та недоліків аналогічних систем, були визначені необхідні вимоги до системи. Встановлені вимоги були розділені на функціональні та нефункціональні.

Серед функціональних вимог особливо були виділені ті, які зосереджені на вирішенні проблем аналогічних систем. Так, наявність авторизації користувача у системі дозволить вирішити проблеми зі збереженням даних та безпекою. Інші вимоги пов'язані з основним функціоналом системи. Система повинна надавати можливість перегляду теоретичних матеріалів, виконання практичних завдань, проходження тестувань, отримання за них оцінок та їх подальшого збереження. Крім

того, система має не тільки зберігати дані користувача, але й надавати можливість їх перегляду. Також, важливими вимогами є зручний інтерфейс та наявність українського перекладу.

З нефункціональних вимог основними були виділені такі, які пов'язані з якісним виконанням функціоналу системи. Основними параметрами оцінки в розроблюваній системі були обрані час виконання та правильність даних. Система повинна обробляти запити швидко і безпомилково. Відправлення повідомлення для про сповіщення про авторизацію користувачів, генерація умов для завдань, перевірка тестувань, відображення даних користувачів, підключення до функціоналу системи, повинні бути виконані за оптимальний час, визначений не меншим, ніж одну секунду.

3 СЦЕНАРІЇ ВИКОРИСТАННЯ СИСТЕМИ

Сценарії використання є важливою частиною проектування системи. Вони визначають поведінку програмного забезпечення при його взаємодії з користувачем і зосереджені на виконанні певної мети. Визначення прецедентів системи допоможуть встановити чіткі алгоритми поведінки системи при різних діях користувача [6]. Таким чином, вже на стадії проектування буде в деталях сформовані функції розроблюваної системи.

На основі вже визначених функціональних вимог були визначені та описані сценарії використання. Детальні зв'язки між ними та акторами зображені на діаграмі прецедентів у додатку А.

3.1 Опис прецедентів системи

Відповідно до визначених в попередньому розділі вимог, були встановлені основні актори системи:

- авторизований користувач;
- неавторизований користувач.

Головним актором для взаємодії з системою є авторизований користувач. Він отримує доступ до основного функціоналу системи після реєстрації нового облікового запису, або авторизації в системі за допомогою вже існуючого. Після здійснення авторизації, користувач може переглядати список наявних кодів, ознайомитися з теоретичним розділом, виконати практичні тренувальні вправи, пройти тестування, ознайомитися зі своїми досягненнями у системі тощо.

Неавторизований користувач не може взаємодіяти з функціоналом системи, поки не створить новий обліковий запис, або не авторизується в системі використовуючи існуючий. Детальний опис кожного прецеденту зображено на таблицях 3.1 - 3.12.

Таблиця 3.1 – Опис прецеденту реєстрації.

Назва	Реєстрація користувача
ID	1
Опис	Користувач вводить свої персональні дані для їх подальшого збереження в системі і створення облікового запису.
Актори	Неавторизований користувач
Організаційні переваги	Користувач створив обліковий запис в системі і має доступ до її функціоналу.
Причина виникнення	Користувач бажає створити новий обліковий запис і натискає кнопку реєстрації.
Передумова	Користувач не має облікового запису, або вийшов з існуючого і представлений меню реєстрації.
Постумова	Користувач успішно здійснив авторизацію з використанням нового облікового запису і отримав повідомлення на електрону пошту.
Головний шлях виконання	1. Користувач вводить необхідні дані і натискає кнопку підтвердження. 2. Система перевіряє правильність введених даних 3. Системою встановлено, що користувач не має існуючого облікового запису з такою електронною адресою. 4. У разі підтвердження, користувач успішно зареєстрований.
Альтернативний шлях виконання	1. Користувач вже має обліковий запис, зареєстрований на ту саму електрону адресу і отримує сповіщення про це. 2. При невірно введених даних, система виведе повідомлення про помилку і запропонує користувачу ввести їх ще раз. 3. Користувач може припинити процес реєстрації достроково, натиснувши кнопку відміни. Обліковий запис не буде створено.
Виключення	При виникненні помилки під час реєстрації, користувач буде сповіщений про це спеціальним повідомленням.

Таблиця 3.2 – Опис прецеденту авторизації користувача.

Назва	Авторизація користувача
ID	2
Опис	Користувач входить в систему використовуючи свої персональні дані облікового запису.
Актори	Неавторизований користувач
Організаційні переваги	Користувач авторизований в системі і має доступ до її функціоналу.
Причина виникнення	Користувач бажає отримати доступ до функціоналу системи і натискає кнопку авторизації
Передумова	Користувач вийшов з існуючого облікового запису і представлений меню авторизації.
Постумова	Користувач успішно здійснив авторизацію з використанням свого облікового запису і отримав доступ до функціоналу системи.
Головний шлях виконання	1. Користувач бажає скористатися функціоналом системи. 2. Користувач вводить необхідні для авторизації дані і натискає кнопку підтвердження. 3. Система перевіряє відповідність введених даних з існуючими обліковими записами. 4. У разі підтвердження, користувач успішно авторизований в системі.
Альтернативний шлях виконання	1. При невірно введених даних, система запропонує користувачу ввести їх ще раз. 2. Користувач може припинити процес авторизації достроково, натиснувши кнопку відміни. Авторизація не буде здійснена
Виключення	При виникненні помилки під час авторизації, користувач буде сповіщений про це спеціальним повідомленням.

Таблиця 3.3 – Опис прецеденту виходу з облікового запису.

Назва	Вихід з облікового запису
ID	3
Опис	Користувач здійснює вихід з системи і свого облікового запису.
Актори	Авторизований користувач
Організаційні переваги	Користувач отримує можливість увійти до системи знову.
Причина виникнення	Користувач бажає вийти з системи і здійснити вхід знову, або з іншого облікового запису.
Передумова	Користувач авторизований в системі.
Постумова	Користувач успішно здійснив вихід з системи і представлений меню авторизації.
Головний шлях виконання	1. Користувач бажає вийти з існуючого облікового запису. 2. Користувач натискає кнопку виходу з системи. 3. Користувач підтверджує свій вибір в діалоговому вікні. 4. Система знищує пов'язані з сесією користувача дані. 5. Користувача представлено меню авторизації.
Альтернативний шлях виконання	Користувач вибирає відміну виходу з системи в діалоговому вікні, процес скасовано, користувач продовжує користуватися системою.
Виключення	При виникненні помилки під час виходу з облікового запису, користувач буде сповіщений про це спеціальним повідомленням і перенаправлений на сторінку авторизації.

Таблиця 3.4 – Опис прецеденту переходження на сторінку вибраного коду.

Назва	Переходження на сторінку вибраного коду
ID	4
Опис	Користувач переходить на сторінку вибраного коду.
Актори	Авторизований користувач

Продовження таблиці 3.4.

Організаціні переваги	Користувач отримує доступ до переліку функціоналу системи, пов'язаного з вибраним кодом, і може розпочинати роботу.
Частота використання	100%
Причина виникнення	Користувач бажає отримати доступ до функціоналу пов'язаного з вибраним кодом.
Передумова	Користувач авторизований в системі і знаходиться на сторінці зі списком наявних кодів.
Постумова	Користувач успішно здійснив перехід і тепер знаходиться на сторінці вибраного коду.
Головний шлях виконання	1. Користувач вибирає потрібний код зі списку. 2. Користувач натискає ссилку переходу. 3. Користувач переходить на відповідну сторінку.
Альтернативний шлях виконання	Альтернативні шляхи виконання відсутні.
Виключення	При виникненні помилки під час переходу на іншу сторінку, або помилки отримання даних, користувачу буде представлено спеціальне повідомлення, і він буде перенаправлений на попередню сторінку.

Таблиця 3.5 – Опис прецеденту ознайомлення з теоретичними відомостями.

Назва	Ознайомлення з теоретичними відомостями
ID	5
Опис	Користувач переглядає теоретичний розділ вибраного коду для вивчення теоретичних знань та прикладів кодування і декодування.
Назва	Ознайомлення з теоретичними відомостями
ID	5

Продовження таблиці 3.5.

Опис	Користувач переглядає теоретичний розділ вибраного коду для вивчення теоретичних знань та прикладів кодування і декодування.
Актори	Авторизований користувач
Організаційні переваги	Користувач отримав доступ до теоретичних матеріалів, може вивчити їх і приступити до розв'язку практичних завдань.
Частота використання	95%
Причина виникнення	Користувач бажає ознайомитися з теоретичними відомостями та прикладами роботи кодування і декодування.
Передумова	Користувач авторизований в системі і знаходиться на сторінці вибраного коду.
Постумова	Користувач успішно перейшов на вкладку з теоретичними відомостями.
Головний шлях виконання	1. Користувач бажає ознайомитися з теорією кодеку. 2. Користувач натискає на посилання для переходу на вкладку з теоретичними відомостями відповідного коду. 3. Користувач переходить на розділ теорії.
Альтернативний шлях виконання	Альтернативні шляхи виконання відсутні.
Виключення	При виникненні помилки під час переходу на іншу сторінку, або помилки отримання даних, користувачу буде представлено спеціальне повідомлення, і він буде перенаправлений на попередню сторінку.

Таблиця 3.6 – Опис прецеденту виконання практичного завдання.

Назва	Виконання практичного завдання
ID	6

Продовження таблиці 3.6.

Опис	Користувач виконує практичні завдання з вибраного коду для тренування, або під час проходження тестування. Завдання має згенерованої умови, проте користувач може запропонувати власну, але лише для тренування.
Актори	Авторизований користувач
Організаційні переваги	Користувач виконує практичні вправи, або проходить тестування, виконуючи основний функціонал системи.
Причина виникнення	Користувач бажає виконати практичне тренувальне завдання, або під час проходження тестування.
Передумова	Користувач авторизований в системі, знаходиться на сторінці вибраного коду і знаходиться на вкладці практиці, або тестування.
Постумова	Користувач виконав вправу і заповнив поле відповіді.
Головний шлях виконання	1. Користувач натискає на кнопку активації завдання. 2. Система генерує умову для завдання 3. Користувач вирішує завдання і заповнює поле відповіді. 4. Після перевірки завдання користувач отримає сповіщення.
Альтернативний шлях виконання	1. Користувач може вибрати свою умову для завдання. Така опція доступна лише під час тренувальних вправ. 2. При невірній відповіді, користувач може спробувати вирішити завдання знову. Така опція доступна лише під час тренувальних вправ. 3. Якщо користувач відповів невірно, і не бажає продовжити виконання завдання, його буде повернено на вікно практики. Така опція доступна лише для тренувальних вправ.
Виключення	При виникненні помилки під час обчислення правильної відповіді, користувачу буде представлено повідомлення про помилку, завдання не буде зараховане.

Таблиця 3.7 – Опис прецеденту проходження тестування.

Назва	Проходження тестування
ID	7
Опис	Користувач проходить тестування для перевірки здобутих навичок.
Актори	Авторизований користувач
Організаційні переваги	Користувач проходить тестування, виконуючи основний функціонал системи
Причина виникнення	Користувач бажає пройти тестування для перевірки знань і натискає кнопку активації.
Передумова	Користувач авторизований в системі, знаходиться на сторінці вибраного коду з активною вкладкою тестування.
Постумова	Система згенерувала відповідні завдання, користувач завершив їх вирішення за заданий час і отримав оцінку.
Головний шлях виконання	1. Користувач натискає на кнопку активації тестування. 2. Система генерує відповідну кількість завдань. 3. Користувач вирішує завдання і заповнює поля відповідей. 4. Система перевіряє відповіді надані користувачем, видає результат і відповідну оцінку. 5. Система зберігає результати пройденого тестування.
Альтернативний шлях виконання	1. Якщо користувач не завершив виконання тестування, система запропонує йому на вибір оцінити зроблені завдання, або перепройти тест знову. 2. Користувач може достроково припинити виконання тестування, у такому випадку, спроба не буде зарахована і оцінена.
Виключення	При виникненні помилки під час оцінювання результатів тестування, користувачу буде запропоновано перепройти тест. Спроба у такому разі не буде зарахована.

Таблиця 3.8 – Опис прецеденту перегляду оцінки за тестування.

Назва	Перегляд оцінки за тестування.
ID	8
Опис	Користувач може ознайомитися з отриманою за пройдене тестування оцінкою в будь-який час на вкладці відповідного тестування, за умови, що дане тестування було успішно пройдене.
Актори	Авторизований користувач
Організаційні переваги	Користувач отримує інформацію про виконане тестування
Частота використання	100%
Причина виникнення	Користувач бажає ознайомитися з отриманою за пройдене тестування оцінкою.
Передумова	Користувач авторизований в системі, знаходиться на сторінці вибраного коду і знаходиться на вкладці з пройденим тестуванням.
Постумова	Користувач успішно перейшов на вкладку з тестуванням і ознайомився з відповідною оцінкою.
Головний шлях виконання	1. Користувач натискає на посилання для переходу на вкладку з пройденими тестуваннями. 2. За умови, що дане тестування було успішно пройдене, користувач ознайомиться з оцінкою.
Альтернативний шлях виконання	1. Користувач не пройшов вибране тестування, тому не побачить результату на сторінці з тестуванням.
Виключення	При виникненні помилки під час переходу на іншу сторінку, або помилки отримання даних, користувачу буде представлено спеціальне повідомлення, і він буде перенаправлений на попередню сторінку.

Таблиця 3.9 – Опис прецеденту перепроходження тестування.

Назва	Перепроходження тестування
ID	9
Опис	Користувач проходить тестування знову для покращення оцінки.
Актори	Авторизований користувач
Організаційні переваги	Користувач перепрохдить тестування для отримання кращого результату.
Причина виникнення	Користувач бажає покращити оцінку за пройдене тестування і натискає кнопку перепроходження на вкладці вибраного тестування.
Передумова	Користувач авторизований в системі, знаходиться на сторінці вибраного коду, знаходиться на вкладці тестування та пройшов дане тестування хоча б раз.
Постумова	Користувач успішно виконав згенеровані системою завдання, пройшов тестування, отримав нову оцінку, яка була збережена
Головний шлях виконання	1. Користувач натискає кнопку для перепроходження тестування на відповідній вкладці. 2. Користувач проходить тестування 3. Якщо користувач успішно завершив тестування, він отримає нову оцінку, яка буде збережена у списку пройдених тестів.
Альтернативний шлях виконання	1. Якщо користувач не завершив тестування за відведений час, система запропонує йому оцінити зроблені завдання, або перепройти тест знову. 2. Якщо користувач не завершив тестування і вийшов з системи, спроба не буде зарахована і оцінена. 3. Якщо користувач отримав гірший результат, він все одно буде збережений і доступний для перегляду у списку з пройденими тестуваннями.

Продовження таблиці 3.9.

Виключення	При виникненні помилки під час оцінювання результатів тестування, користувачу буде запропоновано перепройти тест. Спроба у такому разі не буде зарахована.

Таблиця 3.10 – Опис прецеденту перегляду статистики пройдених тестувань.

Назва	Перегляд статистики пройдених тестувань
ID	10
Опис	Користувач переглядає список результатів усіх тестів, які він проходив.
Актори	Авторизований користувач
Організаційні переваги	Користувач отримує доступ до своїх минулих досягнень в системі
Причина виникнення	Користувач бажає переглянути результати пройдених тестувань і натискає на кнопку активації.
Передумова	Користувач авторизований в системі і знаходиться на сторінці даних користувача
Постумова	Користувач успішно ознайомився з результатами своїх тестувань.
Головний шлях виконання	1. Користувач натискає на кнопку відображення результатів тестувань. 2. Система завантажує результати і відсортовує їх, залежно від дати проходження 3. Користувач отримує доступ до своїх оцінок.
Альтернативний шлях виконання	Альтернативні шляхи виконання відсутні.
Виключення	При виникненні помилки під час завантаження результатів тестувань, користувачу буде відображено повідомлення про помилку, і він буде перенаправлений на попередню сторінку.

Таблиця 3.11 – Опис прецеденту перегляду кількості виконаних завдань

Назва	Перегляд кількості виконаних завдань
ID	11
Опис	Користувач переглядає кількість виконаних тренувальних вправ з вибраного кодеку.
Актори	Авторизований користувач
Організаційні переваги	Користувач отримує доступ до результатів виконання своїх практичних завдань, таким чином, тренування не є марним.
Причина виникнення	Користувач бажає отримати інформацію про виконані тренувальні вправи.
Передумова	Користувач авторизований в системі, знаходиться на сторінці кодеку, успішно виконав хоча б одну тренувальну вправу.
Постумова	Користувач успішно отримав доступ до результатів виконання практичних завдань.
Головний шлях виконання	1. Користувач натискає на вкладку з практикою. 2. Користувач переходить на сторінку тренувальних вправ. 3. Користувач отримує необхідну інформацію.
Альтернативний шлях виконання	Альтернативні шляхи виконання відсутні.
Виключення	При виникненні помилки під час переходу на іншу сторінку, або помилки отримання даних, користувачу буде представлено спеціальне повідомлення, і він буде перенаправлений на попередню сторінку.

Таблиця 3.12 – Опис прецеденту перегляду сторінки довідки.

Назва	Перегляд сторінки довідки
ID	12
Опис	Користувач переглядає додаткову інформацію, яка винесена на окрему сторінку.

Продовження таблиці 3.12.

Актори	Авторизований користувач
Організаційні переваги	Користувач ознайомлений з інформацією про систему, а також зі принципами роботи з функціоналом.
Частота використання	80%
Причина виникнення	Користувач бажає ознайомитися з довідкою і натискає на відповідне посилання.
Передумова	Користувач авторизований в системі, знаходиться в головному меню системи
Постумова	Користувач успішно перейшов на сторінку довідки, та ознайомився з відповідною інформацією.
Головний шлях виконання	1. Користувач натискає на посилання для переходу на сторінку довідки. 2. Користувач переходить на сторінку довідки. 3. Користувач отримує необхідну інформацію.
Альтернативний шлях виконання	Альтернативні шляхи виконання відсутні.
Виключення	При виникненні помилки під час переходу на іншу сторінку, або помилки отримання даних, користувачу буде представлено спеціальне повідомлення, і він буде перенаправлений на попередню сторінку.

3.2 Висновки до розділу

В даному розділі було розглянуто визначення сценаріїв використання розроблюваної системи. Були встановлені неавторизовані і авторизовані користувачі як основні актори системи. Неавторизований користувач описує гостьовий профіль користувача, який не має доступу до основного функціоналу систему і може лише зареєструватися або авторизуватися. Після виконання цих дій, гість стає

авторизованим користувачем, який отримує доступ до основних прецедентів.

З основних сценаріїв використання можна виділити реєстрацію, авторизацію, перегляд теоретичного розділу, виконання тренувальних практичних вправ, проходження тестування, та перегляду власних досягнень у системі: пройдені тестування, кількість спроб, оцінки, кількість вірно виконаних практичних завдань. Під час детального опису кожного прецеденту, були встановлені основні, бажані шляхи його виконання, а також і альтернативні. Це допоможе забезпечити обробку виникаючих помилок і необхідну гнучкість функціоналу при реалізації програмного забезпечення.

4 ВИБІР ТА ОБГРУНТУВАННЯ ЕЛЕМЕНТІВ ТА ТЕХНОЛОГІЙ

Перед тим як розпочати етап розробки програмного забезпечення, необхідно розглянути існуючі рішення, проаналізувати найбільш підходящі технології і вибрати ті, які забезпечать якісну та швидку реалізацію системи. Важливо встановити платформу розробки, вид програмного забезпечення, яке буде реалізовувати систему, мову програмування та основні фреймворки та бібліотеки.

4.1 Обгрунтування архітектури системи

Як було розглянуто в попередніх розділах, основні функціональні вимоги системи можуть бути реалізовані в будь-якому представленні. Для даної системи були розглянуті реалізації у вигляді мобільного додатку, настільного застосунку, та веб-застосунку. Розглянемо кожний, для визначення яка платформа підходить для розроблюваної системи.

Настільний застосунок має свої переваги, проте їх куди менше, ніж недоліків. Користувач може встановити настільний застосунок, або завантажити архів і розпакувати його на локальній машині. Система відразу готова до використання. Застосунок зручно використовувати, проте, на додачу до тих недоліків, які були розглянуті під час аналізу аналогів, має ряд недоліків, пов'язаних з його розробкою. Потрібно забезпечити кросплатформеність додатку, що накладає обмеження на технології, які можна використовувати при розробці. Також, необхідно забезпечити захищеність даних, що проблематично зробити для повністю офлайн додатку [7]. Недобросовісні користувачі можуть підробити інформацію про пройдене тестування, використовуючи спеціальні програми. Щоб забезпечити цілісність даних, найбільш доцільним буде використання окремого API, яке буде надавати доступ до даних, які будуть зберігатися зовні програми. Це в свою чергу призведе до ряду проблем пов'язаних з поширенням даних між модулями системи. Також, при обмежені системи настільним застосунком, додання до неї нових модулів і її подальше розширення може бути проблематичним.

Мобільний додаток має схожі недоліки. Для забезпечення надійності даних необхідно зберігати їх окремо від додатку, і використовувати для захищення окреме API, а вимога кросплатформенності накладає обмеження на технології для реалізації. Також, додаток на мобільному пристрої хоч і є портативним, і доступний в будь-якому місці, обмеження розміру екрану не дозволять виконувати повний функціонал системи.

Веб-застосунок, на відміну від опцій, розглянутих вище, дозволяє в певній мірі вирішити перераховані проблеми. Користувач отримує доступ до функціоналу веб-застосунку через браузер, тобто таким чином досягається кросплатформеність системи. Якщо в попередніх випадках використання Інтернету було вимушене, то в цьому виправдане і вирішує ряд недоліків. Важливі дані захищені від зловмисних користувачів, і не можуть бути змінені, шляхом доступу до пам'яті локальної машини. Звісно, дані, які відправляються від клієнта до сервера можуть бути перехоплені і пошкоджені. Проте оригінальні дані зберігаються на сервері, і можуть бути використані для відновлення [8].

Також, перевагою використання веб-застосунку є гнучкіша архітектура, ніж у настільного додатку. Така система дозволяє розбиття на окремі модулі, перенести логіку виконання програмного забезпечення на серверну частину, представлення на клієнтську. Такий підхід дозволяє розподілити функціонал системи на мікросервіси, що надасть ще більшу гнучкість архітектури. Система матиме можливості для обслуговування і розширення [9, 10].

Ще однією перевагою є можливість розподілення функціоналу на серверну та клієнтську частину забезпечує легкість портування системи на інші платформи. Маючи реалізовану серверну частину, яка містить бізнес-логіку системи, створити різні клієнти для різних платформ буде доволі легко. Оскільки такий підхід не прив'язується до конкретного представлення, система може мати декілька клієнтів, які використовують функціонал одного серверу. Таким чином, користувач, використовуючи один обліковий запис, зможе виконати тестування на настільному додатку для операційної системи Windows, переглянути результати на мобільному застосунку під Android, а потім змінити свої дані через веб-сайт.

Для реалізації клієнта було вирішено використовувати односторінковий додаток. Односторінковий додаток – це такий вид веб-застосунку або сайту, що взаємодіє з користувачем шляхом динамічного завантаження необхідної інформації на сторінку замість того, щоб завантажувати усю сторінку повністю з серверу. Такий підхід зменшує переривання при переході зі однієї сторінки на іншу. Це робить застосунок схожим на додаток для персональних комп'ютерів. В односторінковому додатку увесь необхідний код, наприклад HTML, JavaScript і CSS завантажуються один раз разом з усією сторінкою, або необхідні ресурси завантажуються динамічно по мірі необхідності.

Сторінка ніколи не перезавантажується, ні надає контроль до іншої сторінки, проте користувач може вільно переходити на інші сторінки за допомогою історії, створюючи ілюзію багатьох розділів додатку.

Такий підхід дозволяє зменшити навантаження на серверну частину, що дозволить оптимізувати роботу системи. Клієнт буде звертатися до серверу лише для оновлення окремих елементів сторінки, або здійснення запитів для отримання, збереження і редагування необхідної інформації. Враховуючи визначені для розроблюваної системи вимоги, односторінковий додаток є найбільш підходящим рішенням [11].

4.2 Вибір веб-фреймворку

Для реалізації веб-застосунку, особливо важливим є вибір підходящого веб-фреймворку. На сьогоднішній день, існують десятки різних інструментів для розробки веб-застосунків. Кожен з них призначений для вирішення певних проблем у сфері веб-розробки, тому необхідно визначити найбільш підходящий, враховуючи перераховані вище вимоги. Для аналізу і порівняння були обрані найбільш популярні інструменти.

jQuery – це швидка, невелика та багатофункціональна бібліотека JavaScript. Бібліотека дозволяє легко отримати доступ до будь-якого елементу DOM, обробляти події, звертатися до атрибутів та змісту елементів веб-сторінки та маніпулювати

ними. Також, jQuery надає зручне API для роботи з Ajax [12].

Перевагами jQuery є легкість її використання під час розробки. Зазвичай бібліотека підключається в файл розмітки і готова до використання. При необхідності додаткового функціоналу бібліотеки, підключаються додаткові плагіни. Це дозволяє зібрати для проекту саме ту необхідну функціональність JavaScript. Також наявна можливість підключення бібліотеки через менеджер пакетів, наприклад npm.

З недоліків jQuery можна виділити наступні. При підключення js-файлів бібліотеки до кожної веб-сторінки, яка її використовує, збільшується час відображення сторінки, що знижує продуктивність всього веб-застосунку. Цьому можна запобігти, використовуючи CDN, кешування, проте ці методи також мають свої недоліки. Плагін для користувацького інтерфейсу jQuery має доволі скудний функціонал. Також, відсутнє внутрішнє управління залежностями і плагінами, наприклад обумовлене завантаження сценаріїв виконання [13].

Отже, бібліотека jQuery, хоч і має свої привабливі сторони, не підходить для розробки даної системи. Оскільки функціонал системи буде реалізовано на мові JavaScript, підключені великі js-файли, яких не модна розділити, будуть зменшувати час виконання кінцевого продукту, що суперечить встановленим вимогам. Бібліотека jQuery підходить для простих анімацій, зміни стилів, ajax-запитів та маніпуляції з елементами для сторінки. Але для реалізації складного веб-застосунку, одного лише jQuery недостатньо.

Веб-фреймворк Angular був признаний одним із найбільш популярних інструментів для розробників за результатами опитування StackOverflow 2018 [14]. Такий високий рейтинг обумовлений зростанням популярності односторінкових додатків, і фреймворк Angular, переписаний з AngularJS, був орієнтований на їх розробку. Хоча Angular вважається JavaScript фреймворком, рекомендується використання TypeScript, синтаксичною надбудовою над JavaScript, яка підтримує статичну типізацію.

Найбільшою перевагою фреймворку є його архітектура, заснована на компонентах. На відміну від AngularJS, з його MVC архітектурою, Angular-

застосунок розподіляється на незалежні логічні та функціональні компоненти. Ці компоненти можуть бути замінені, об'єднані, або використані в інших частинах застосунку. Така незалежність робить тестування легким і забезпечує безпомилкову роботу кожного компонента.

Компілятор, який використовує Angular і багато інших фреймворків, конвертує вихідний код і HTML у простий JavaScript під час збірки. Код компілюється перед тим як браузер завантажить веб-застосунок, таким чином сторінка відображається швидше. Також, однією з переваг Angular є його інтерфейс командної строки. Він автоматизує процеси ініціалізації застосунку, його налаштування і конфігурації за допомогою простих команд.

Angular також надає інструменти для впровадження залежностей. Ці залежності визначають як різні компоненти взаємодіють і показує як зміни в одній частині коду впливає на інші частини. Впровадження залежностей робить код читабельним і захищеним. Проте, процес визначення залежностей між компонентами може зайняти багато часу.

З недоліків фреймворку можна виділити слабку SEO-оптимізацію, яка пов'язана з тим як односторінкові застосунки змінюють контент та теги сторінки використовуючи JavaScript. Також, Angular має доволі важку інфраструктуру, яка містить безліч різних функцій, які можуть бути зайвими в більш легкому проекті. Архітектура фреймворку є складною, розгалуженою і більше орієнтована на великі застосунки. Більше того, архітектура обмежує гнучкість вибору інших технологій при розробці, що може бути значним недоліком при використані підходу мікросервісів [15].

Іншим популярним інструментом для розробки веб-застосунків є JavaScript бібліотека React. Вона орієнтована на створення односторінкових додатків та мобільних застосунків, оскільки найбільш підходить для отримання і маніпулювання даними, які постійно змінюються. Перевагою React є його відносна легкість. При необхідності, складні веб-застосунки використовують додаткові бібліотеки для управління станом додатку, маршрутизації, та взаємодії зі сторонніми API.

Як і Angular бібліотека React використовує структуру засновану на

компонентах, яка правда є простішою. Компоненти React можуть бути використані багаторазово і зазвичай описують кнопки, списки та інші. Ці компоненти обернені в компоненти-обгортки, і так до кореневого компонента. Така структура полегшує відображення компонентів на веб-сторінці, розробку застосунку та його тестування.

Ще однією перевагою React є використання віртуального DOM. Зазвичай веб-застосунки містять велику кількість інтеракції з користувачами. Застосунок приймає дані, редагує їх, відображає зміни. Оновлення DOM потребує великих затрат ресурсів, і при частому оновленню, це може стати серйозним недоліком для продуктивності застосунку. Тому, React використовує віртуальний DOM, який зберігає зміни, а потім порівнює минулий і теперішній стан елементів, і вираховує кращий спосіб для їх зміни без оновлення усього DOM.

Проте, бібліотека має наступні недоліки. Користувацький інтерфейс компонентів описується за допомогою JSX, декларативного XML-подібного синтаксису, який працює с JavaScript. Такий підхід має свої переваги, але він обмежує написання коду під React. Такий шаблон не можна використати в інших застосунках.

Також, не зважаючи на широку і поширену екосистему, деякі компоненти не підтримуються на рівні з основною бібліотекою, що може вчинити проблеми при розробці. Особливим недоліком є обмеженість інтерфейсу командної строки, яка не дозволяє робити поглиблене налаштування проекту [16].

Фреймворк Vue набрав велику популярність. Vue легко інтегрується в проекти з використанням JavaScript-бібліотек, але й може функціонувати як веб-фреймворк для розробки односторінкових додатків. Vue перейняв багато сильних сторін інших фреймворків, зокрема React. Vue має гнучку засновану на компонентах архітектуру, використовує віртуальний DOM. Як і React, Vue має багату екосистему для забезпечення маршрутизації, управлінням стану тощо. Проте, Vue офіційно підтримує компоненти своєї екосистеми і постійно синхронізує їх до останньої версій основної бібіліотеки. На відміну від Angular, Vue можна підключити до окремої частини існуючого проекту використовуючи CDN, або мінімізований js-файл [17].

Компоненти, які використовує Vue, кожен містить HTML-шаблон, CSS стилі, та JavaScript код. Таким чином, частини користувацького інтерфейсу можна імпортувати в інший застосунок, або використати пре-процессори.

Проте Vue має і ряд недоліків. Одним з яких є недостатня підтримка TypeScript. Зважаючи на орієнтованість Vue на невеликі розміри і загалом легку архітектуру, не завжди використання усіх можливостей типізованого TypeScript є доцільним. Проте, у випадках, коли це необхідно, його підтримка залишає бажати кращого. Структура компонентів, яка змішує HTML, CSS та JS в одному файлі, може швидко стати нечитабельною[18]. Також, у Vue відсутні чіткі архітектурні шаблони. Це особливо помітно при роботі з менеджером стану Vuex. В документації нема чіткого плану, де саме реалізовувати логіку застосунку, в компонентах, окремих модулях чи в менеджері стану.

Отже, після аналізу існуючих фреймворків, було прийнято рішення використати Vue. На відміну від Angular, він легкий і простіший в використані, має усі кращі сторони бібліотеки React. Vue також швидко розвивається, має підтримку для сторонніх бібліотек. Одним із найбільших переваг для розроблюваної системи є широкі можливості для розширення додатку та додання нових модулів. Використовуючи архітектуру компонентів, яку надає Vue, систему легко можна розширити новими модулями. Також було вирішено використати компоненти екосистеми Vue: Vuex, для централізованого управління станом застосунку, VueRouter, для маршрутизації застосунку, конфігурації параметрів шляху та контролю навігації та VueResource, для роботи з веб-запитами.

4.3 Вибір мови розробки

Оскільки розроблювану систему було вирішено реалізовувати в якості веб-застосунку на Vue, були розглянуті дві мови програмування – JavaScript і TypeScript. Vue надає можливість розробляти додатки з використанням обох мов, тому розглянемо кожну, визначимо яка з них більше підходить для реалізації системи.

Основною мовою програмування для клієнтської частини є JavaScript. Разом з

HTML і CSS, JavaScript є одним із основних інструментів для побудови веб-сайтів і використовується для створення інтерактивних і динамічних сторінок. JavaScript підтримує створення об'єктів на прототипі і цим відрізняється від більшості традиційних мов, заснованих на класах. Він може функціонувати як процедурна, так і об'єктно-орієнтована мова. Таким чином, розробник може використовувати JavaScript по різному, залежно від поставлених задач [19]. Гнучкість і функціональність JavaScript, що узгоджується з Vue.js, тому його використання є цілком аргументованим.

Vue також надає підтримку TypeScript. Проте, його інтеграція не підтримується офіційно, синтаксис в компонентах є громіздким. На відміну від JavaScript, який інтерпретується під час виконання, TypeScript висвітлює помилки компіляції під час розробки, що зменшує шанс виникнення помилок під час виконання програми. TypeScript забезпечує строгу типізацію об'єктів, що дозволяє перевірити правильний тип змінної під час компіляції. Отже, якщо TypeScript є кращою версією JavaScript, чому не використовувати його? Враховуючи недостатню підтримку TypeScript у Vue, яка не буде покращена до виходу найближчої версії, його використання для розробки даної системи не є доцільним.

Більше того, Vue має компоненту структуру, а компоненти побудовані у вигляді анонімних літералів об'єктів, тому більшість сильних сторін TypeScript була би просто зайвою, а його компіляція в простий JavaScript може збільшити час виконання і зменшити продуктивність системи. Тому, зважаючи на визначені раніше вимоги, JavaScript цілком задовольняє основні з них, тому був обраний в якості мови програмування.

4.4 Інші технології

4.4.1 Firebase

Для взаємодії з серверною частиною, було вирішено використовувати VueResource, яка є складовою екосистеми Vue. Оскільки, більша частина функціоналу системи пов'язана з клієнтською частиною, для виконання поставлених

умов є можливість перекласти роботу серверної частини, використовуючи такий інструмент як Firebase.

Firebase надає широкий набір компонентів, які можуть бути використані для заміни функціоналу серверної частини. Одними з таких є база даних в реальному часі, система авторизації, аналітики та інші. Перевагою Firebase є те, що його можна використовувати як для веб-застосунку, так і для мобільних додатків. Головною причиною використання його в даній системі є доволі низька кількість навантаження на серверну частину. В даній системі, основними задачами серверу є авторизація користувача і збереження даних про тестування і пройдені завдання. Firebase здатний піклуватися про це і надає набагато більше інструментів. Крім того, не потрібно турбуватися про масштабування, продуктивність сервера та розмір бази даних, Firebase виконує масштабування автоматично. Ще однією перевагою даного сервісу є можливість використати його для хостингу веб-застосунку [20].

4.4.2 Бібліотека компонентів

Для розробки елегантного, приємного та інтуїтивного користувацького інтерфейсу, існують багато популярних бібліотек, які пропонують готові елементи користувацького інтерфейсу. Найбільш популярною з них є Bootstrap. Bootstrap надає готові CSS стилі, які підключаються до проекту через CDN, і можуть бути використаними для конкретних елементів. Використання бібліотек на зразок Bootstrap, дозволяє швидко і ефективно розробляти користувацький інтерфейс.

Оскільки ми використовуємо фреймворк Vue.js, було вирішено використати бібліотеки, розроблені спеціально для нього. З найбільш популярних можна виділити:

- Vue Material;
- Material Components Vue;
- Vuetify.

Кожна з вище перерахованих бібліотек легко інтегрується в готовий проект, використовуючи CDN, за допомогою менеджеру пакетів, або в якості плагіну за

допомогою інтерфейсу командної строки Vue [21]. Проте, кожна бібліотека має певні обмеження.

Material Components Vue – використовує підхід Material Design від Google, і використовує доволі прості компоненти Ця бібліотека має найменшу кількість готових елементів користувацького інтерфейсу на вибір. На відміну від інших варіантів, вона не має інтеграції з Nuxt.js, що робить візуалізацію на сервері значно важчою.

Vue Material має великий перелік готових компонентів. Її особливістю є наявність компонентів спеціально розроблених для екосистеми Vue, наприклад для VueRouter, який має кілька своїх компонентів. Також, бібліотека має широкий набір різних конфігурацій, які дозволяють ефективно імпортувати і використовувати цілі теми для користувацького інтерфейсу.

Vuetify найбільш повна і найбільш складна бібліотека з вище перерахованих. Вона охоплює широкий спектр різноманітних елементів користувацького інтерфейсу, починаючи з випадаючого списку і закінчуючи більш абстрактними такими як слайд-шоу. Кожен компонент Vuetify містить гнучку логіку і багато параметрів налаштування, для того щоб підстроїти використання компонента для окремих випадків [22]. Бібліотека активно розвивається, і незважаючи на відсутність офіційної підтримки з боку Vue, її функціонал охоплює усі необхідні вимоги розроблюваної системи, тому в якості Vuetify було обрано в якості бібліотеки компонентів.

4.4.3 Інструменти для роботи з модулями

Під час розробки веб-застосунку використовується велика кількість сторонніх бібліотек, фреймворків та пакетів. Для забезпечення їх правильної взаємодії, необхідно використати відповідний менеджер пакетів. Такий інструмент контролює версії пакетів, їх встановлення та використання. Для розроблюваної системи використання менеджеру пакетів є необхідним. На даний момент найбільш популярними є npm і yarn. Не дивлячись на популярність і поширеність npm, yarn

має файл блокування, має вищу швидкість встановлення пакетів, і вони усі автоматично зберігаються у package.json. Проте, npm є найбільш надійним інструментом. Менеджер npm має велику онлайн базу даних загальнодоступних та платних приватних пакетів, яка називається реєстром npm. Реєстр доступний через клієнта, а наявні пакети можна переглядати та шукати через веб-сайт npm. Також, офіційна документація Vue рекомендує використовувати npm, тому було використовувати даний менеджер пакетів.

Для того, щоби збирати пакети і ресурси, якими управляє npm, необхідний збірник модулів, який буде компілювати файли пакетів в JavaScript модулі для їх подальшого підключення у веб-застосунок. Для даної системи був обраний Webpack. Webpack є найбільш популярним рішенням для цієї задачі. Він може бути налаштований під необхідні задачі. Однією з переваг даного збірника є можливість динамічного імпортування, яке зручно співпрацює з Vue [23]. Використовуючи динамічне імпортування, є можливість розбити компоненти Vue на логічні модулі, які будуть завантажені при першому використанні. Таким чином, компоненти будуть завантажені при необхідності, що покращить продуктивність всього застосунку.

4.4.4 Система контролю версій

Використання системи контролю версій є необхідністю під час розробки програмного забезпечення. Можливість відстежувати поступову реалізацію окремих компонентів системи перетворює лінійний процес розробки в графік, коли в будь-який момент часу можна повернутися на кілька кроків назад, для виправлення помилок, або відмовитися від внесених змін. Навіть не використовуючи їх для командної розробки, вони все одно являються важливим інструментом для написання програмного забезпечення. Для даної системи був обраний Git. Дана система контролю версій найбільш поширена, має зручний інтерфейс командної строки, можливість безкоштовного хостингу веб-сторінок, та створення приватних репозиторіїв без обмежень.

4.5 Висновки до розділу

В даному розділі були проаналізовані існуючі технологічні рішення для практичної реалізації системи. Було вирішено розробляти систему в якості веб-затосунку. Дане рішення має значні переваги над іншими варіантами, а також має потенціал для розширення іншими клієнтами.

Для реалізації веб-застосунку були проаналізовані наявні веб-фреймворки і бібліотеки, на основі яких були обрані всі інші технології. Для аналізу були обрані jQuery, Angular, React та Vue. Після аналізу було встановлено, що для даної системи найбільше підходить Vue, через його легкість, практичність, швидкість візуалізації компонентів та орієнтованість на відносно невеликі проекти. Враховуючи архітектуру Vue, його слабку підтримку TypeScript, та відсутність необхідності у строгій типізації, було вирішено обрати JavaScript в якості основної мови програмування. Для зручної побудови користувацького інтерфейсу були розглянуті найпоширеніші бібліотеки компонентів Vue такі як Vue Material, Material Components Vue, та Vuetify. З даного переліку був обраний останній, оскільки Vuetify має найбільший перелік готових компонентів і має можливості для їх детального налаштування.

Як було встановлено в попередніх розділах, основна частина бізнес-логіки зосереджена на клієнтській частині, серверна виконує операції авторизації та діставання і запису даних. Тому, було прийнято рішення використати Firebase, що дозволить перенести функціонал серверної частини на сторонні сервіси, і зосередитись на самому додатку і користувачах. Так було вирішено використати безсерверну архітектуру. В якості інструментів для роботи з внутрішньою екосистемою, були обрані менеджер пакетів npm і збірник модулів Webpack. Обидва є найбільш поширеними інструментами для своїх завдань відповідно, і добре співпрацюють з фреймворком Vue.js.

5 СТРУКТУРНА СХЕМА СИСТЕМИ

Розробка структурної схеми програми являється одним із найважливіших етапів в процесі розробки програмного забезпечення. Структурна схема описує сукупність усіх компонентів системи, зв'язки і методи обміну інформацією чи іншими об'єктами між ними. Спроектована структурна схема дозволить легко модифікувати програмне забезпечення при необхідності, якщо зміни обґрунтовані покращенням якості.

Під час побудови структурної схеми було вирішено розподілити систему на кілька модулів. Модулі можуть відрізнятися від один одного розміром, реалізацією, кількістю застосованих компонентів, але у контексті структурної схеми, було прийнято рахувати їх в якості однакових частин системи, які містять певну логіку функціоналу. Структурна схема розроблюваної системи представлена на рисунку 5.1.

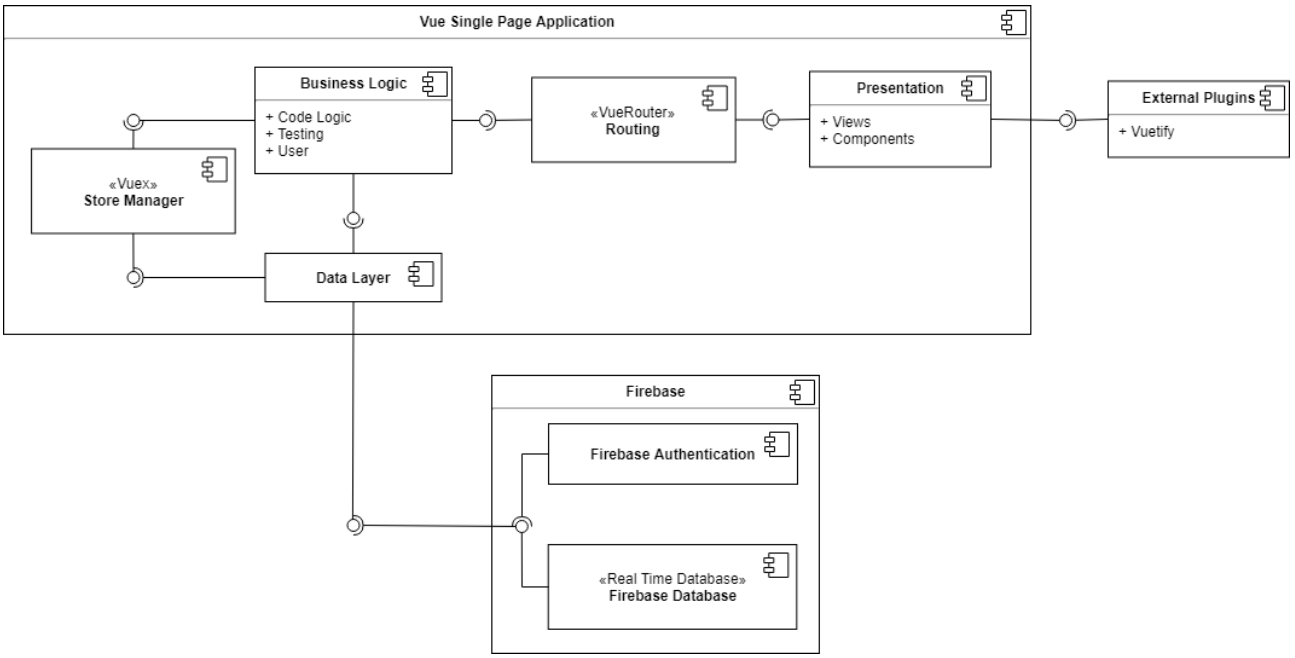

Рисунок 5.1 – Структурна схема системи

Як зображено на рисунку, систему можна поділити на дві частини: односторінковий додаток Vue.js та базу даних Firebase. База даних містить наступні компоненти:

- Firebase Authentication.
- Firebase Real Time Database.

Односторінковий додаток в свою чергу складається з:

- Data Layer.
- Store Manager.
- Presentation.
- Business Logic.

Також, система взаємодіє зі сторонніми бібліотеками для представлення, які позначені як External Plugins.

5.1 Компонент Firebase

Враховуючи визначені вимоги та сценарії використання, в якості сховища даних було вирішено використати Firebase. Така база даних взаємодіє з односторінковим додатком напряму. Для забезпечення авторизації користувача в системі використовується модуль Firebase Authentication, а для збереження інформації – Frebase Real Time Database. База даних слугує сховищем облікових записів користувачів, їх персональних даних, оцінок за виконані тестування, пройдені завдання, тощо. Структура компоненту Firebase зображена на рисунку 5.2.

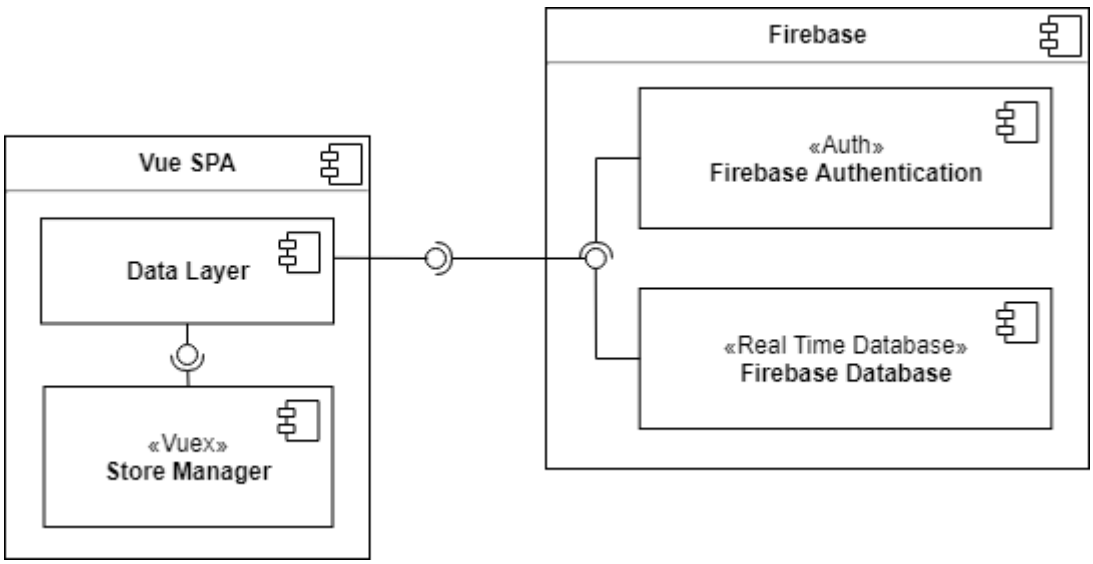

Рисунок 5.2 – Взаємодія компоненту Firebase з системою

Взаємодію з базою даних виконують компоненти для роботи з даними. Для даних загального призначення використовується компонент Data Layer. Він взаємодіє з базою даних, та дістає потрібну інформацію, використовуючи протокол HTTPS. Задача Store Manager полягає в зберіганні даних, які необхідні для роботи багатьом компонентам застосунку протягом усієї сесії, наприклад користувача, і поширення цих даних. З Store Manager та Data Layer будуть працювати більшість модулів системи.

5.2 Компонент Vue SPA

Односторінковий додаток можна поділити три умовних розділи:

- компоненти бізнес-логіки;
- компоненти представлення;
- компонент маршрутизації;
- компоненти для роботи з даними та станом.

Структура компоненту зображена на рисунку 5.3.

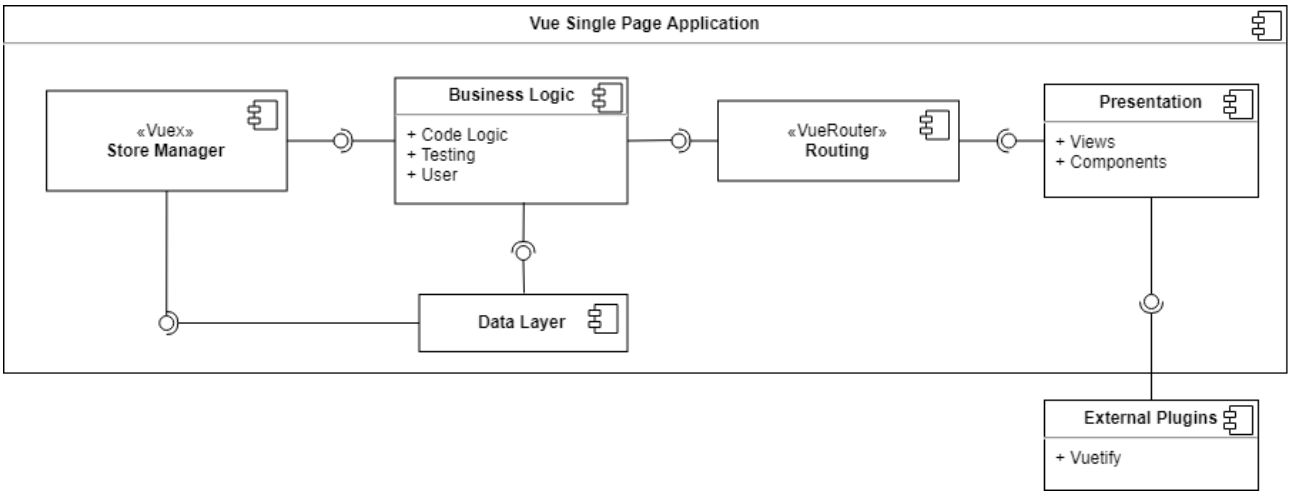

Рисунок 5.3 – Компонент Vue SPA

Компоненти Data Layer та Store Manager відповідають за зв'язок з джерелом даних та поширенням інформації до інших частин додатку.

З цими компонентами співпрацює модуль Business Logic. Цей модуль містить

реалізацію усієї бізнес-логіки додатку, а також відповідає за правильне представлення інформацію з рівня даних.

Модуль Business Logic складається з трьох окремих компонентів: Code Logic, Testing та User.

Реалізація роботи кодеків зосереджена в компоненті Code Logic, причому кожний кодек представлений в якості окремого модуля. Таким чином, зберігається гнучкість системи, і можливість реалізації нових кодеків без порушення роботи застосунку. Компонент Code Logic широко використовується іншими модулями системи.

Компонент Testing містить реалізацію функціоналу генерування завдань, компонування тестувань та їх перевірку. Цей компонент взаємодіє з компонентами Code Logic та Data Layer. Code Logic використаний для створення завдань та їх перевірки, а через Data Layer результати зберігаються в базі даних.

Компонент User відповідає за логіку роботи з даними, які стосуються користувача, отримання оцінок, пройдених тестувань та інші. Компонент User взаємодіє з Store Manager, який зберігає інформацію про користувача та розширює її в додатку.Дані користувача, пов'язані з пройденими завданнями і тестуваннями будуть зберігатися у базі даних.

Модуль Routing описує налаштування різних шляхів застосунку і відповідних веб-сторінок. Він слугує своєрідним зв'язуючи модулем між компонентами бізнес-лоігки та представлення.

Модуль Presentation містить функціонал для відображення користувацького інтерфейсу, власне самі веб-сторінки та компоненти, з яких вони складаються. Для готових елементів користувацького інтерфейсу було використано бібліотеку компонентів Vuetify, яка підключається до додатку в якості модуля External Plugins. Для маршрутизації додатку використаний плагін VueRouter. Рівень представлення буде здійснювати запити для отримання необхідної інформації, наприклад, кількість виконаних користувачем завдань, і представляти їх в коректній формі.

5.3 Висновки до розділу

В даному розгляді було описана побудова структурної схеми для розроблюваної системи. Для системи були визначені основні компоненти і модулі. Основною складовою системи є односторінковий додаток, який містить більшу частину логіки системи. Він складається з компонентів для роботи з даними та стану, компонентів бізнес-логіки, маршрутизації та представлення. Окрім цих, він також використовує сторонні плагіни, зокрема бібліотеку Vuetify для покращення рівня представлення. Бізнес-логіка поділена на три логічні компоненти. Компонент Code Logic містить реалізацію кодування і декодування для кожного з представлених кодів, компонент Testing містить функціонал для генерації тестів та завдань, а компонент User реалізує логіку для роботи з даними, які стосуються користувача, наприклад отримання оцінок та пройдених тестувань.

Функції серверу і бази даних виконує Firebase. Цей модуль можна поділити на два компоненти – авторизації та бази даних. Firebase забезпечує та автоматизує процеси авторизації та збереження даних. Односторінковий додаток зв'язується з базою даних за протоколом HTTPS, використовуючи більш універсальний та загальний компонент Data Layer, та Store Manager, функція якого полягає в збереженні та поширені даних, які використовуються по всьому додатку.

Таким чином, була досягнута гнучка архітектура, яка дозволить розширювати систему при необхідності.

6 РОЗРОБЛЕННЯ ER-ДІАГРАМИ

В попередніх розділах, зважаючи на встановлені вимоги, було обрано Firebase Real Time Database в якості бази даних.

Дані в Firebase зберігаються нереляційно, в хмарному сховищі і оновлюються у режимі реального часу. Замість постійних HTTP запитів використовується синхронізація даних. Таким чином, усі підключені до бази даних клієнти отримують доступ до одного екземпляру сховища даних і автоматично отримують оновлення з новою інформацією. Це доволі зручно, при поширені даних між багатьма користувачами, зміни одного зможуть відразу побачити усі. Для даної системи ця функція використовується для обрахунку таймерів на виконання тестування. Збереження цієї інформації в реальному часі забезпечить її правильність та цілісність. Ще однією зручною функцією бази даних Firebase є можливість доступу до інформації офлайн, через локальне збереження даних, які синхронізуються при відновлення з'єднання [24].

Уся інформація в базі даних зберігається нереляційно, в форматі JSON, у вигляді деревоподібної структури. На відміну від SQL баз даних, Firebase не має таблиць або записів. При додаванні даних до JSON дерева, вони стають вузлами в існуючій структурі з відповідним ідентифікатором. У такому вигляді їх легко використовувати на клієнтській стороні, запити є простішими і швидшими. Оскільки для даної системи відсутні складні структури, такий формат є підходящим.

Враховуючи визначені для системи сценарії використання, було встановлені наступні сутності:

- User;
- CompeletedTest;
- ActiveTest;
- TestTask;
- PracticeRecord;

Загальна структура бази даних зображена в додатку Б.

Сутність User описує обліковий запис користувача системи. Вона містить

загальну інформацію, таку як ім'я та групу, інформацію, необхідну для авторизації в системі, а також статистику роботи користувача: пройдені тестування та вправи. Поля сутності описані в таблиці 6.1.

Таблиця 6.1 – User

Назва поля	Тип даних	Опис
id	String	Унікальний ідентифікатор
email	String	Електрона адреса користувача, використовується при авторизації.
name	String	Ім'я користувача, зберігається для загальної інформації.
group	String	Група користувача, зберігається для загальної інформації.
completedTests	Array	Виконані тести, зберігаються посилання на об'єкти.
completedPractice	Array	Записи виконаних тренувальних вправ.

Сутність CompeletedTest описує пройдений тест, який зберігається для статистики. Сутність містить отриманий результат та час, за який було виконано тестування. Також, зберігається дата завершення, за якою сортуються виконані тести. Поля сутності описані в таблиці 6.2.

Таблиця 6.2 – CompletedTest

Назва поля	Тип даних	Опис
id	String	Унікальний ідентифікатор
userId	String	Унікальний ідентифікатор користувача, який зберігається для зв'язку з об'єктом користувача
name	String	Назва кодеку з якого було пройдено тестування, зберігається для загальної інформації.

Продовження таблиці 6.2

Назва поля	Тип даних	Опис
codecCode	String	Код відповідного кодеку, з якого було пройдене тестування.
result	Number	Результат пройденого тестування, зберігається для оцінювання.
timeUsed	String	Час, затрачений на виконання тестування, зберігається для загальної інформації.
Date	Date	Дата завершення тестування, зберігається для загальної інформації.

Сутність ActiveTest описує тестування, яке в даний момент проходить користувач. Воно відрізняється від вже пройденого тим, що зберігає згенеровані завдання і таймер, який відраховує час, затрачений на виконання тестування. Зберігання активного тесту в реальному часі дозволить зробити роботу таймера безперервною та безпомилковою. Після виконання тестування, активний тест зберігається в базі як виконаний. Поля сутності описані в таблиці 6.3.

Таблиця 6.3 – ActiveTest

Назва поля	Тип даних	Опис
Id	String	Унікальний ідентифікатор
userId	String	Унікальний ідентифікатор користувача, який зберігається для зв'язку з об'єктом користувача
Name	String	Назва кодеку з якого проходиться тестування, зберігається для загальної інформації.
codecCode	String	Код відповідного кодеку, з якого проходиться тестування.
timeLeft	Number	Час, який залишився для виконання тестування, зберігається в реальному часі.

Продовження таблиці 6.3.

Назва поля	Тип даних	Опис
timeTotal	Number	Загальний час, виділений на виконання тестування.
tasks	Array	Згенеровані системою завдання для даного тестування.

Сутність TestTask описує практичне завдання, згенероване для тестування. Сутність зберігає повідомлення, яке необхідно закодувати або декодувати, та список параметрів, необхідних для виконання. Поля сутності описані в таблиці 6.4.

Таблиця 6.4 – TestTask

Назва поля	Тип даних	Опис
id	String	Унікальний ідентифікатор
testId	String	Унікальний ідентифікатор тестування, якому належить завдання, зберігається для зв'язку з батьківським об'єктом.
message	String	Згенерована умова завдання
params	Array	Додаткові параметри, необхідні для виконання завдання.

Сутність PraticeRecord описує статистику виконаних вправ. Вона зберігає кількість розв'язаних завдань з певного кодеку. Поля сутності описані в таблиці 6.5.

Таблиця 6.5 – PracticeRecord

Назва поля	Тип даних	Опис
id	String	Унікальний ідентифікатор
userId	String	Унікальний ідентифікатор користувача, який виконав дане завдання.

Продовження таблиці 6.5.

Назва поля	Тип даних	Опис
codecCode	String	Код відповідного кодеку, з якого була виконана тренувальна вправа.
amount	Number	Кількість виконаних користувачем тренувальних вправ.

В даному розділі було описано використану нереляційну, хмарну базу даних Firebase Real Time Database. Її особливістю є оновлення даних у реальному часі, використовуючи синхронізацію даних з підключеними клієнтами, що було використано для збереження таймеру під час виконання тестування. Firebase Real Time Database локально дублює дані, завдяки чому система може функціонувати офлайн. Дані у базі даних зберігаються у форматі JSON дерева. Такий підхід найкраще підходить для нескладних сутностях веб-застосунків – дані не потребують додаткової обробки для використання на клієнті, що додатково підвищує швидкість виконання запитів до бази даних.

Сутності, які використовуються в роботі системи, були описані в якості вузлів на деревоподібній структурі бази даних Firebase. З основних можна виділити User, CompletedTest, ActiveTest, TestTask, PracticeRecord. Вони описують обліковий запис користувача в системі, виконане тестування, активне тестування, завдання та записи тренувальних вправ. Всі сутності мають прості, не рекурсивні відношення між собою. Це дозволяє зробити базу даних розширюваною та легко масштабованою.

7 РЕАЛІЗАЦІЯ БІЗНЕС-ЛОГІКИ

7.1 Архітектура односторінкового додатку

Після проведеного аналізу вимог, вибору технологій та побудови структурної схеми, було прийнято рішення використати безсерверну архітектуру і реалізувати клієнтську частину в якості односторінкового додатку. Для подальшої реалізації було розроблено архітектуру додатку.

Оскільки односторінковий додаток взаємодіє з користувачем динамічно, він підгружає необхідну інформацію на сторінку, замість того, щоб завантажувати усю сторінку з серверу. Фреймворк Vue.js робить розробку односторінкових додатків зручнішим і компактнішим. Однією з причин є розподілена архітектура додатків Vue.js, яка заснована на компонентах.

Кожен компонент описується в окремому .vue файлі, і містить HTML-шаблон, який описує частину веб-сторінки, Vue.js об'єкт, який описує бізнес-логіку компоненту, і CSS стилі, які застосовуються до шаблону.

Точкою старту односторінкового додатку є сторінка index.html, яка містить один блоковий елемент з визначеним id. Головними функціями index.html є збереження мета-даних сторінки, підключення сторонніх бібліотек та файлів через CDN, та визначення блокового елементу. Фреймворк Vue.js створить кореневий екземпляр Vue, до якого вбудовуються усі плагіни та частини еко-системи Vue.js. Кореневий екземпляр єдиний об'єкт Vue, який створюється використовуючи ключове слово new і описаний у окремому файлі main.js. В main.js конфігурується кореневий екземпляр та імпортуються плагіни.

Завдяки кореневому екземпляру, фреймворк Vue.js, використовуючи id елементу, відобразить замість блокового елементу кореневий компонент App.vue. HTML-шаблон компоненту буде відображено на сторінці. Кореневий компонент відрізняється від інших, оскільки він включає у себе усі інші компоненти, і не може бути відтвореним.

Заснована на компонентах архітектура дозволяє створити деревовидну структуру додатку, яку зручно розширювати, додавати нові модулі, видаляти,

використовувати їх знову. Проте, при розширенні додатку, і збільшені кількості модулів, такий підхід має недоліки. Оскільки вся архітектура складається з компонентів, підтримка кожного стає все складнішим. Тому у фреймворку Vue, використовується розподілення компонентів по спеціальним пакетам [25].

Окремим пакетом знаходяться усі плагіни, які використовуються в односторінковому додатку. Зазвичай, усі плагіни імпортуються, конфігуруються та використовуються у файлі main.js. Це є зручним, при невеликій кількості використовуваних плагінів, оскільки всі настройки і імпортування знаходиться в одному місці. Проте при збільшення їх кількості, файл стає засміченим і не читабельним. Оскільки при розробці даної системи було вирішено використовувати значну кількість плагінів, а саме Vuetify, Vuex, VueRouter та VueResource, було практичним винести файли з їх конфігурацією в окремі пакети. При додавання нового плагіну з використанням vue-cli, відповідний файл налаштування для буде створений автоматично, і доданий до пакету plugins. Проте, для великих модулів екосистеми, наприклад Vuex та VueRouter, правильнішим буде створити окремі пакети store та router відповідно.

В односторінковому додатку Vue прийнято розділяти усі компоненти представлення на дві категорії – компоненти які представляють власне веб-сторінки, і компоненти, які описують елементи користувацького інтерфейсу на цих сторінках. Вони зберігаються у двох різних пакетах views та components відповідно. Такий розподіл дозволяє легко визначити компонентну ієрархію, оскільки компоненти з views складаються з більш дрібних компонентів з components.

Також, компоненти, які описують повну веб-сторінку, використовуються при маршрутизації додатку у конфігурації VueRouter. Таким чином кожному шляху відповідає одна сторінка, тобто один компонент.

Компоненти, які знаходяться у пакеті cormponents, зазвичай є допоміжними і описують окремий елемент користувацького інтерфейсу, наприклад панель навігації, або випадаючий список. Для даної системи була використана бібліотека компонентів Vuetify, більшість необхідних елементів користувацького інтерфейсу вже реалізовано. Проте, для таких компонентів, які широко використовуються в усьому

додатку створено окремий пакет ui, який є дочірнім пакетом components.

Таким самим є пакет layout, який містить компоненти, які описують різні частини веб-сторінки, наприклад HomeFooter.vue, HomeNav.vue. Таким чином, призначення цих компонентів є зрозумілим, і масштабування, а також пітдтримання проекту є простішим.

Усі інші компоненти, які належать до пакету components, відносяться до дочірнього пакету codes. Вони описують інтерфейс і логіку представлення для сторінок з кодами. Ці компоненти розділені по реалізованим кодам і належать до відповідних пакетів. Окремі компоненти реалізують сторінку теоретичних відомостей, практичних завдань на тестування. Використовуючи динамічне імпортування, ці компоненти автоматично імпортуються на активну сторінку і доступні для перегляду користувачем. Таки чином, зберігається можливість розширення системи новими кодами. Для включення нових модулів, необхідно додати реалізацію в відповідний пакет та реалізувати компоненти для кожного елементу коду – теоретичних даних, практичних завдань і тестування. Система автоматично перевірить доступний реєстр кодів і включить нові як доступні до використання. Основна частина бізнес-логіки належить до пакету codeLogic. В ньому знаходяться js-файли відповідних кодів, які містять логіку кодування і декодування. Вони можуть імпортуватися в компоненти для багаторазового використання логіки. Таким чином компоненти кодеків містять лише логіку представлення, а функціонал кодування і декодування залишається виділеним окремо.

Пакет auth містить логіку, пов'язану з авторизацією користувача в системі. Компоненти пакету здійснюють валідацію введених даних, відправляють запити на серверну частину, зберігають авторизованого користувача сесії у менеджері стану додатку. Для фреймворку Vue.js не прийнято поєднувати шаблони представлення та логіку, особливо коли вона безпосередньо взаємодіє з зовнішніми сервісами, наприклад авторизації, та не стосується власне компоненту. Тому функціонал авторизації винесений в окремі файли і належить окремому пакету. Структуру модулів системи зображено на рисунку 7.1. Більш детально структура компонентів

додатку та їх взаємодія зображені у додатках В і Г.

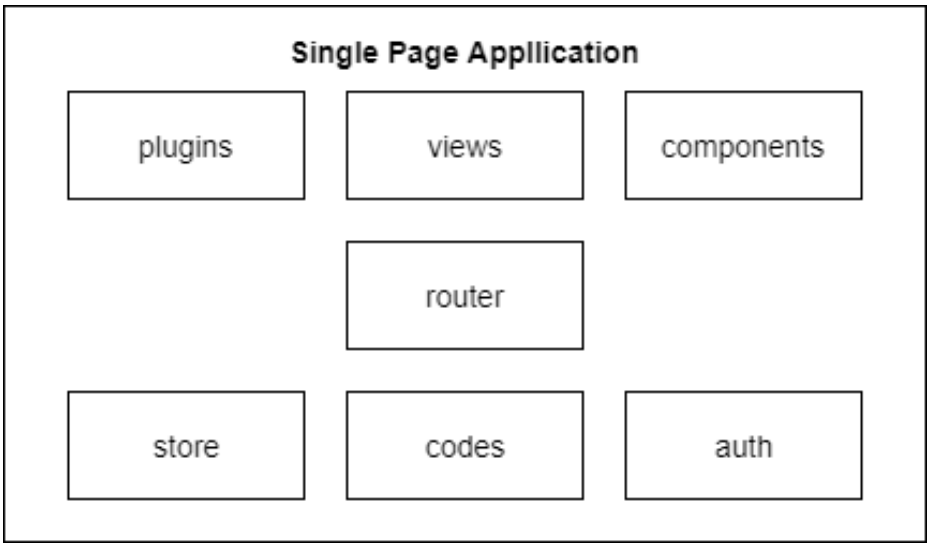

Рисунок 7.1 – Структура модулів пакетів застосунку

7.2 Використані шаблони проектування

Для якісної та зручної розробки розподіленої та масштабованої системи необхідне використання шаблонів проектування. Шаблони надають готові рішення для розв'язання проблем з написанням програмного забезпечення і є важливими для розробки великих продуктів. Для реалізації даної системи, були обрані наступні шаблони проектування.

7.2.1 Шаблон проектування MVVM

Шаблон MVVM розроблений як модифікація шаблону PresentationModel, та орієнтований на відокремлення користувацького інтерфейсу від розробки бізнес логіки. Таким чином відбувається чітке розподілення системи, що дозволяє розподілити відповідальність на різні частини додатку [26]. Шаблон MVVM проектування містить три складові:

- View, що представляє собою користувацький інтерфейс додатку. В даній системі представлений компонентами представленнями, які знаходяться у пакетах

views та components;

- Model, що представляє собою власне модель, дані, які необхідні для роботи додатку. Для даної системи модель представлена даними користувача, кодеків та тестувань, які зберігаються в базі даних;

- ViewModel, що представляє собою абстракцію представлення, а також обгортку даних, для зв'язку між моделлю і представленням. Також ViewModel відслідковує зміни в даних, що зроблені користувачем, а також відпрацьовує логіку роботи View.

Шаблон MVVM підходить для додатків, в яких наявне зв'язування даних, а оскільки було прийнято рішення застосувати фреймворк Vue.js, його використання для даної системи є цілком виправданим.

Веб-фреймворк Vue.js орієнтований на частину ViewModel шаблону і зберігає його сутність. Використовуючи двосторонній зв'язок даних, Vue.js поєднує модель і представлення. Кожен об'єкт Vue є ViewModel, частиною що об'єднує дві інші частини шаблону.

Для окремих компонентів, рівнем представлення слугує сам DOM, або його елементи, які управляються екземпляром Vue. Кожен об'єкт асоційований з відповідними елементом веб-сторінки, таким чином, Vue.js використовує шаблонізацію засновану на DOM. При створенні Vue екземпляра, під час ініціалізації додатку, або під час створення компонента, він рекурсивно обходить усі дочірні вузли елемента, до якого він прив'язаний і встановлює необхідні зв'язки з даними. Після компіляції, представлення отримує можливість реагувати на зміни моделі.

Таким чином, відпадає необхідність в маніпулюванні з елементами DOM напряму. Під час змін даних оновлення представлення буде відбуватися автоматично за допомогою об'єкту Vue. Елементи представлення оновлюються дуже точно, аж до окремого вузла. Таким чином збільшується продуктивність односторінкового додатку, оскільки змінюється лише окремі частини, а не вся сторінка.

Моделлю шаблону у Vue.js слугує модифікований анонімний JavaScript об'єкт,

або об'єкт даних. Ці об'єкти належать компонентам Vue і описуються в якості функцій, які повертають об'єкт даних. Таким чином, забезпечується унікальність моделі кожного компонента, в протилежному випадку, усі компоненти посилалися би на один і той самий об'єкт даних.

При створенні об'єкта Vue, і його об'єкта даних, екземпляр створює для кожного поля об'єкта відповідні властивості, які стежать за змінами і сповіщають екземпляр Vue про необхідність оновлення. Таким чином, модель стає реактивною. Завдяки Vue відпадає необхідність перевіряти наявність змін вручну, а також самостійно оновлювати представлення. Більш детально сутність шаблону у окремих компонентах зображена на рисунку 7.2.

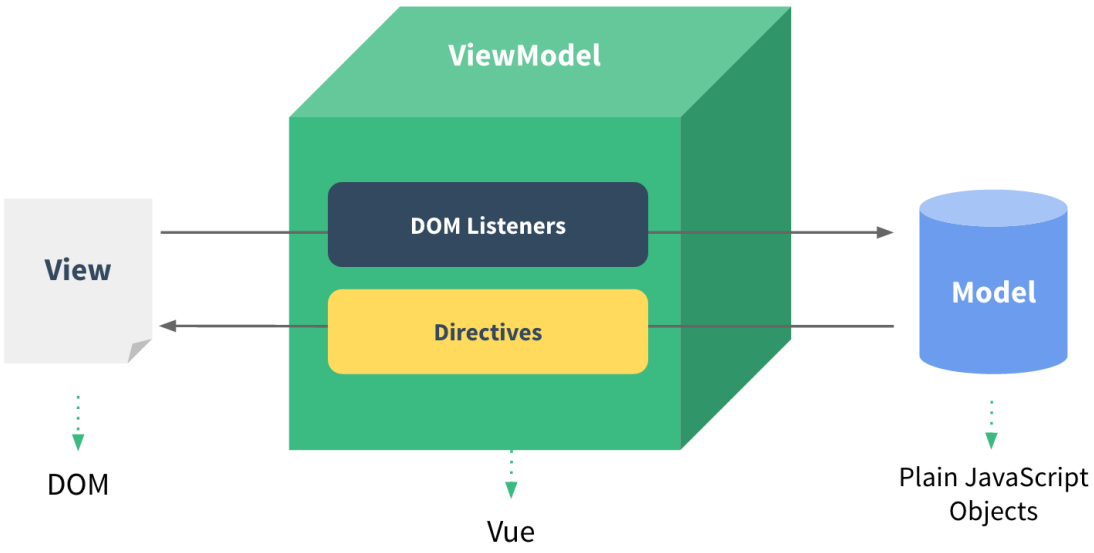

Рисунок 7.2 – Шаблон MVVM у Vue.js [27]

Таким чином, кожен компонент Vue реалізує шаблон проектування MVVM, використовуючи двостороннє зв'язування даних та слухачів подій. Створюється наступна система: представлення оновлюється при зміні моделі, а модель реагує на зміни представлення, де Vue виступає посередником. А оскільки Vue.js використовує компонентну архітектуру, кожен дочірній елемент стає частиною рівня ViewModel.

7.2.2 Шаблони реалізації односторінкових додатків Vue.js

При реалізації даної системи були використані необхідні шаблони реалізації які рекомендовані для додатків Vue.js [28]. Ці підходи до розробки спрямовані на покращення якості написаного коду та полегшенні роботи з багатьма модулями системи. З основних можна виділити:

- конкретне призначення компонентів;
- модульний підхід;
- стан додатку;
- стилі компонентів.

Оскільки уся архітектура односторінкового додатку Vue.js побудована на компонентах, для розробки великої і складної системи необхідно щоби компоненти були реалізовані з певними вимогами. Загальний підхід до розробки визначає, що більшість компонентів повинні бути багаторазовими, реалізовувати конкретну задачу та бути незалежними і не прив'язаними до архітектурного рішення.

Це означає, що компонент повинен створюватися задля певної мети і не повинен зсередини звертатися до стану додатку або до сторонніх ресурсів. Якщо ми додамо таку логіку до компонента він, по-перше, стає великим в обсязі, а, по-друге, втрачається весь сенс застосування багаторазових компонентів.

Таким чином, для даної системи було використано модульний підхід. Замість накопичення бізнес-логіки в компонентах, які повинні відповідати за логіку представлення користувацького інтерфейса, більша частина функціоналу була винесена в окремі файли, та модулі. Наприклад, уся бізнес-логіка кодування та декодування, авторизації, генерації завдань для тестувань, дій з користувачем, функціонал для роботи з менеджером стану, винесені в окремі js-файли. Це дозволяє зберегти можливість розширення, багаторазового використання і додання нових варіантів реалізації.

Для великих застосунків, в яких обмін даними є складнішим за односторонній обмін між батьківським компонентом та дочірнім, виникає потреба в загальному сховищу даних, до якого мали б доступ усі компоненти додатку. Таке сховище надає

менеджер станів Vuex. Файли налаштування і використання функціоналу менеджеру стану знаходяться в пакеті store. Використання Vuex може призвести до небажаного переводу частини бізнес-логіки додатку з компонентів в менеджер стану. Щоб запобігти цьому, в розроблюваній системі було введено ряд правил для роботи зі станом додатку.

По-перше, усі дані, які зберігаються в Vuex, повинні зберігатися лише в одному місці. Компоненти, такі як UserPage.vue, Code.vue, лише використовують ці дані, вони не створюють власних копій для роботи. Таким чином, збережена цілісність даних. По-друге, в Vuex зберігаються лише ті дані, які необхідно поширити в усьому додатку, для даної системи дані про авторизованого користувача. Компоненти не можуть самостійно змінювати стан додатку, доступ до нього має лише пакет авторизації. Таким чином, використання Vuex не призводить до зміщення усієї бізнес-логіки до менеджеру стану, а навпаки запобігає ускладненню обміну інформацією між компонентами.

Під час розробки клієнтської частити веб-застосунку та збільшенні кількості і об'єму сторінок, існує загроза виникнення великої кількості css файлів. Робота з ними та їх підтримка може створити додаткові складнощі при розробці. Для даної системи було використано бібліотеку компонентів Vuetify, тому кількість css класів є невеликою. Якщо вони необхідні, вони описуються в конкретних компонентах, використовуючи атрибут scoped, для обмеження їх використання. Такий підхід не дозволяє css класи накопичуватися. Для глобальних класів використано окремий глобальний css файл.

7.3 Авторизація користувача з використанням Firebase та Vuex

Реалізація авторизації користувача в системі була виконана з використанням Firebase Authentication. Для поширення даних був використаний менеджер стану Vuex. Логіка авторизації реалізована в менеджері стану та модулі auth.

Реєстрація користувача відбувається в компоненті RegisterForm.vue, з якого складається сторінка реєстрації. Для створення облікового запису в системі,

користувач вводить електрону адресу, повне ім'я, номер групи, заповнює поля паролю та підтвердження паролю. Для полів вводу електронної пошти, паролю та його підтвердження застосована валідація даних. Компоненти полів Vuetify мають вбудований механізм перевірки, який використовує правила, описані у окремому файлі formRules.js. Виділення їх в окремий файл дозволить їх використовувати для будь-якої форми застосунку, позбавляючи від необхідності описувати правила знову. Ці правила перевіряють правильність введеної електронної адреси, відсутність пустих полів, довжину пароля та відповідність поля пароля і його підтвердження, а також описують повідомлення, яке отримує користувач при порушення конкретного правила. Якщо введені дані не узгоджуються цим правилам, користувач отримує відповідне повідомлення, а форма помічається як не валідна.

Ім'я та номер групи зберігаються для загальної інформації, а електрона адреса та пароль використовуються для процесу авторизації. Firebase надає необхідний інструментарій для реалізації авторизації багатьма способами, для даної системи була використана авторизація через електрону адресу та пароль, оскільки Firebase автоматично забезпечує такі функції як відправка повідомлення для підтвердження реєстрації, зміна паролю та інше.

Для коректної роботи додатку, необхідне використання інформації про авторизованого в сесії користувача в різних компонентах. Щоби запобігти ускладненню обміну даних між компонентами, було вирішено перенести частину логіки авторизації до менеджеру стану Vuex. Функціонал менеджеру описаний в модулі store і поділяється на чотири великі функції:

- state;
- getters;
- mutations;
- actions.

State зберігає власне стан додатку, для даної системи це змінна authUser, який описує авторизованого користувача. Getters описують функції доступу до стану, mutations дозволяють змінювати стан додатку, і зазвичай недоступні поза контекстом Vuex, а actions описують логіку роботи менеджеру.

В actions реалізовані методи signUpUser, signInUser та signOutUser, які описують методи реєстрації, авторизації та виходу користувача з системи. Ці методи викликаються в компонентах RegisterForm та LoginForm, використовуючи this.$store. Для реєстрації використовується метод Firebase, який створює користувача, використовуючи вказану поштову адресу та пароль. Таким чином, користувач зберігається в Firebase, йому присвоюється унікальний ідентифікатор, і записується в state використовуючи метод setUser. Для авторизації користувача, використовується метод Firebase, який дістає з бази обліковий запис, використовуючи поштову адресу та пароль, який користувач вводить у компоненті LoginForm.vue. Для позбавлення користувача від необхідності авторизуватися кожний раз, дані користувача зберігаються в localStorage. Логіка цього збереження реалізована в модулі auth.

Використовуючи функції getters з менеджеру стану, інші компоненти можуть використовувати інформації про зареєстрованого користувача. Його використовують AppHeader.vue, для відображення на панелі навігації. UserPage.vue, для відображення інформації користувача та інші. Більш детально процес авторизації в системі зображено на діаграмі послідовності в додатку Д.

7.4 Реалізація логіки кодеків

Як було описано вище, функціонал роботи кодеків було винесено в окремий модуль codeLogic. Таким чином, велика частина коду була відокремлена від логіки компонентів, які описують тестування, або практичні завдання для конкретних кодеків. Окремі файли з потрібними кодеками підключаються і використовуються при необхідності в компонентах.

Модуль codeLogic поділений на чотири дочірніх модуля: systematicCodes, cyclicCodes, nonbinaryCodes та otherCodes, кожен з яких охоплює реалізовані систематичні, циклічні та недвійкові коди відповідно. Такий поділ дозволяє логічно структурувати наявні файли за типом завадостійких кодеків. Кожен дочірній модуль містить js-файли, які описують реалізацію кодування та декодування для кожного кодеку.

Кожен js-файл фактично описує компонент Vue без HTML шаблону. Цей компонент може містити всі властивості, які має екземпляр Vue: дата об'єкт, методи, методи-спостерігачі, обчислювальні властивості. При підключення такого файлу до компоненту в якості спеціальної Vue властивості mixins, всі його властивості стають доступними до цього компоненту і можуть вільно використовуватися всередині Vue екземпляра.

Кожен міксин з модуля codeLogic містять два основних методи: code та decode, які реалізують логіку кодування та декодування коду відповідно. Ці методи зазвичай приймають повідомлення, вже відформатоване та перевірено компонентом, який використовує даний міксин, та допоміжну інформацію, яку використовує кодек для правильної роботи. Наприклад, розмір алфавіту для первинних недвійкових кодів, або складену матрицю для простого ітераційного коду. Також, кожен кодек має методи checkAnswer для перевірки відповідей користувача для кодування і декодування. Цей метод використовує методи code та decode та повертає булеве значення про помилку або правильну відповідь. Деякі коди мають додаткові методи для виконання допоміжних методів. Для кодування деяких кодів необхідне обчислення або використання додаткових інструментів. Тому, кожен файл має своєрідну і особливу структуру і виправдовує функціональний підхід який використовує Vue.js. Більш детально роботу модуля кодування зображено на прикладі коду Бергера на додатку Е.

7.5 Генерація практичних завдань та оцінка тестування

Компоненти, які використовують файли модуля codeLogic, поділені та дві групи: компоненти для практичних завдань та компоненти для тестування користувача, наприклад GreyCode.vue, GreyDecode.vue та GreyTest.vue для коду Грея. Компоненти для практики для кодування та декодування відокремлені один від одного.

Компоненти кодування та декодування описують інтерфейс для виконання конкретного практичного завдання. Зазвичай вони містять поле для зображення

повідомлення, яке потрібне згенерувати, поля для додаткової інформації, при необхідності. Перед відправкою відповіді, користувач має можливість власноруч вибрати повідомлення, яке перевірить система. Якщо повідомлення, введене користувачем відповідає вимогам для даного коду, то вирішене завдання буде зараховано. Також, повідомлення може випадково генеруватися системою, використовуючи вибрані користувачем параметри. Це забезпечується різними елементами користувацького інтерфейсу, як слайдери, випадаючі списки тощо. Це дозволяє налаштувати виконання практичного завдання, для того, щоб користувач міг краще зрозуміти роботу кодеку.

Після підтвердження відповіді користувачем, компонент перевіряє її, використовуючи підключений файл даного кодеку. Для того, щоб виконані тестові вправи мали вагу для користувача, система зберігає кількість правильно виконаних вправ з кожного кодеку. При правильній відповіді, користувач отримає відповідне повідомлення, а його обліковий запис в даних буде оновлено. Для цього робиться окремий запит в базу даних. Транзакції в Firebase захищені спеціальними правилами, які конфігуруються через консоль Firebase. Ці правила забезпечують захищеність підключення і виключають можливість доступу до бази даних зовні застосунку.

Таким чином, користувач буде бачити свій прогрес розвитку при розв'язанні практичних завдань. При неправильній, користувач отримає вибір продовжити виконання завдання, чи скасувати процес вирішення. Якщо користувач скасує процес, то йому буде запропоновано нове завдання та всі налаштовані параметри будуть встановлені за замовчуванням, у протилежному випадку його відповідь і умова буде збережена, і користувач зможе продовжити виконання.

Компоненти тестування мають схожий функціонал з компонентами для практичних завдань. Створюється об'єкт ActiveTest, який описує активне тестування. Кожен тест містить п'ять завдань на кодування та п'ять на декодування. Логіка генерування умов завдань винесена в файл TestGenerate.js, який використовує модуль codeLogic для отримання параметрів складення умов. Завдання, які генеруються компонентом, не відрізняються від тих, що виконував користувач в компонентах з практикою. Проте, на відміну від них, користувач не може змінити

умову та параметри, та після виконання завдання відсутнє повідомлення про правильність. Для кожного кодеку визначений час на виконання тестування, який відраховує спеціальний таймер. При початку тестування, починає він запускається і відмірює час до завершення. Значення таймеру зберігається в компоненті, проте для запобігання ситуації, в якій користувач закриває і відкриває вкладку браузера, таймер тестування зберігається в базі даних. Оскільки Firebase забезпечує збереження даних в реальному часі, значення таймеру в додатку синхронізується з даними в базі даних.

Якщо користувач не закінчить тестування вчасно, він може отримати оцінку за виконані завдання, або розпочати знову. Для кожного кодеку в базі даних зберігається кількість використаних на тестування спроб. Кількість спроб необмежена, проте зберігається для звітності.

Після завершення тестування, система збирає відповіді у масив та послідовно перевіряє їх. Далі, система з об'єкту ActiveTest створює об'єкт CompletedTest, який зберігає результат пройденого тестування, дату проходження та затрачений час. Таким чином, користувач завжди зможе переглянути свої успіхи та при необхідності підтвердити виконання. Архів з історією пройдених тестів та виконаних тренувальних вправ доступний у компоненті UserResults.vue. Більше детально алгоритм виконання та перевірки тестування зображено на додатку Ж.

7.6 Висновки до розділу

В даному розділі була розглянута реалізація бізнес-логіки системи. Система була реалізована в якості безсерверного односторінкового додатку з використанням Vue.js, який має засновану на компонентах архітектуру, то було вирішено розбити компоненти додатку на модулі. Були створені наступні модулі: auth, codeLogic, store, router, components, view.

Для взаємодії між рівнем представлення та даних у компоненті Vue був використаний шаблон проектування MVVM. В якості представлення в даній системі реалізовані компоненти views та components, які описують елементи

користувацького інтерфейсу. В якості моделі використані дані користувача, які зберігаються в базі даних, обліковий запис користувача, дані про тестування та виконані вправи. Зв'язуючою одиницею є Vue екземпляри, які поєднують модель і представлення за допомогою подій та двостороннього зв'язку даних.

Авторизація користувача була реалізована за допомогою сервісу Firebase, менеджеру стану Vuex, та модулю auth. Користувач може зареєструватися та авторизуватися в системі використовуючи електрону адресу та пароль. Дані користувач зберігається в менеджері стану Vuex, через який його інформація доступна усьому додатку, що забезпечує рівний обмін даними між компонентами.

Модуль codeLogic містить реалізовану логіку кодування та декодування кодів. Кожен код реалізований в окремому файлі, які потім підключаються до потрібних компонентів, де їх функціонал використовується для перевірки практичних завдань або тестування. Функціонал генерування завдань реалізований в окремому файлі TestGenerate.vue, який використовується у компонентах тестувань. При проходженні тестування, був реалізований таймер, значення якого зберігається за допомогою Firebase, який забезпечує його безперебійну роботу.

8 РОЗРОБЛЕННЯ КОРИСТУВАЦЬКОГО ІНТЕРФЕЙСУ

Розробка користувацького інтерфейсу є важливою складовою процесу реалізації програмного забезпечення. Зручний дизайн дозволить користувачу легко взаємодіяти з функціоналом системи. Оскільки система призначена для навчального моделювання, наявність простого, зрозумілого, ефективного, приємного інтерфейсу зробить роботу з програмним забезпеченням кращим досвідом.

8.1 Реалізація компонентів додатку

В попередніх розділах було вирішено використати бібліотеку Vuetify для готових компонентів, за специфікацією Material Design. Застосування Vuetify для розробки користувацького інтерфейсу дозволило спростити написання клієнтської частини і зосередитися на написанні логіки представлення застосунку.

Оскільки Vuetify використовує готові шрифти та іконки, необхідні файли були підключені через CDN. Також, Vuetify надає велику кількість власних та заснованих на Bootstrap класів для глибшого налаштування компонентів.

Компоненти, використані при розробці користувацького інтерфейсу, можна поділити на:

- компоненти розмітки та контейнери;
- компоненти для групування елементів;
- компоненти для форм, зчитування та вводу даних;
- компоненти навігації;
- компоненти кнопок та індикатори;
- спливаючі та модальні вікна.

Компоненти розмітки забезпечують положення інших елементів користувацького інтерфейсу на сторінці. Вони використовуються для поділу сторінки на секції. Кожен компонент Vue може містити лише один кореневий елемент і такі компоненти виконують роль обгортки. V-container слугує в якості контейнера загального призначення і не має багатьох властивостей для

налаштування. В даній системі він використовується для групування усієї сторінки в центрі та створення відступів. V-row, v-col та v-flex більш поширені та специфічні за використанням і орієнтовані на невеликі групи елементів інтерфейсу.

Компоненти для групування складають більш вужчу групу ніж контейнери. Вони також дозволяють розміщувати елементи на сторінці, проте на відміну від них, також надають певний готовий функціонал. Найбільш поширеним компонентом є v-card, який фактично описує готову сторінку. V-card багатофункціональний компонент, при розробці даного інтерфейсу його було використано для написання форм авторизації та реєстрації, та компонентів для кодеків. Серед інших можна виділити v-tabs та v-list, які використані для створення динамічної табуляції для практичних завдань кодування і декодування, і для написання списків реалізованих кодеків.

Компоненти форм орієнтовані на відображення полів вводу та виводу. Готові компоненти v-text-field та v-form використані для реалізації форм користувачів. Ці компоненти є надбудовою над аналогічними HTML-елементами, і надають зручні властивості для налаштування їх роботи. Так, для форм вже є готові валідації, які можна перевіряти прямо на клієнті.

Навігаційні компоненти надають користувачу можливість переміщатися по додатку. Компоненти v-menu та v-toolbar слугують контейнерами для посилань на інші сторінки системи та дозволяють зручно групувати їх.

Компоненти кнопок та індикатори описують інтерактивні елементи користувацького інтерфейсу, з якими взаємодіє користувач. До них належать різні види кнопок, полоса завантаження та інші.

Компоненти діалогових вікон використовуються для донесення необхідної інформації до користувача та представлення вибору. Компонент V-dialog описує діалогове вікно, який може слугувати обгорткою для будь-якого іншого компонента. Компонент CodeCompleteDialog.vue використовує V-dialog для побудови модального вікна, яке дозволяє користувачу вибрати продовження дії після відправки відповіді на практичне завдання. За допомогою готових інструментів Vuetify були реалізовані власні компоненти, які складають веб-сторінки додатку.

Усі реалізовані компоненти представлення розділені на пакети views та components, тобто компоненти, які описують веб-сторінки та компоненти, які описують окремі елементи користувацького інтерфейсу. Компоненти веб-сторінок містять:

- HomePage.vue.
- LoginPage.vue.
- RegisterPage.vue.
- UserPage.vue.
- CodePage.vue.
- InfoPage.vue.

Компонент HomePage.vue описує головну сторінку додатку. Він складається з компонентів AppHeader.vue, Home.vue і CodeList.vue. Головне меню додатку зображено на рисунку 8.1.

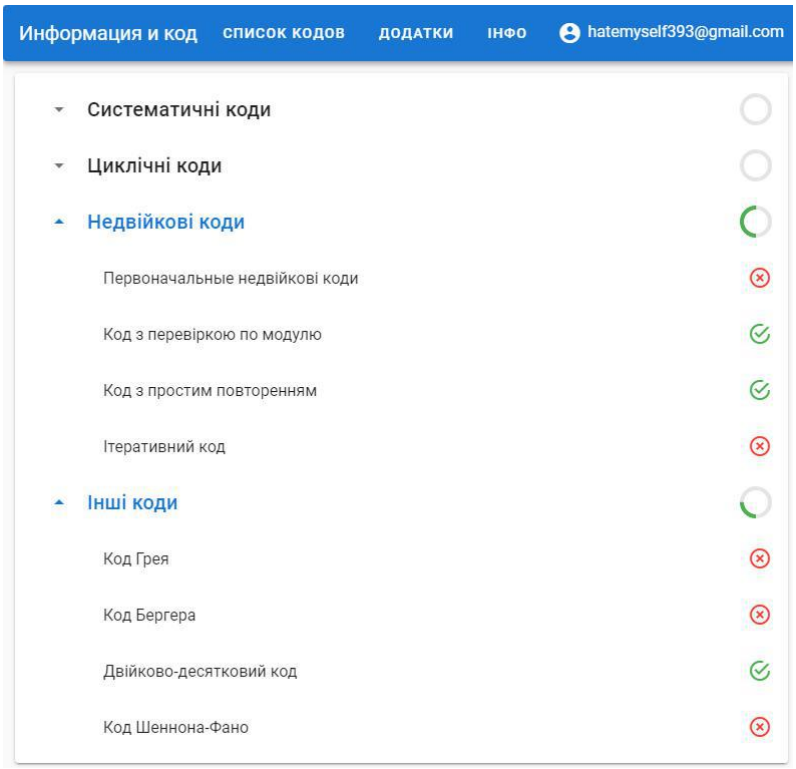

Рисунок 8.1 – Головне меню додатку

Головна сторінка містить панель навігації додатку, яка дозволяє перейти сторінку довідки, перейти на сторінку користувача, продивитися список кодів, або

перейти назад до головної сторінки меню. Також, через панель навігації можна вийти з системи і перейти на сторінку авторизації. Головна сторінка відображає список кодів, які доступні до вирішення користувачем, і зображує прогрес користувача в системі.

Компонент LoginPage.vue описує форму авторизації існуючого облікового запису користувача. Він містить поля для вводу електронної адреси та паролю, та кнопки підтвердження. Поле паролю має спеціальну маску, яку надає Vuetify, і використовується для маскування введеної інформації.

Компонент Register.vue описує форму реєстрації нового облікового запису користувача. За допомогою компонентів v-row та v-card та Bootstrap класів було створено потрібну розмітку сторінки. Для групування полів реєстрації і їх подальшої валідації використано компонент v-form. Компоненти v-text-field дозволяють налаштувати вигляд повідомлення, а також використати властивості rules для валідації. Для полів пароля та підтвердження пароля застосовано спеціальна маска, яка приховує введені символи. Також компоненти v-text-field мають додаткове поле, яке відображає відповідне повідомлення при виникнення помилки під час валідації. Кнопка підтвердження реєстрації заблокована до проходження валідації форми.

Компонент UserPage.vue описує сторінку користувача. Вона складається з компоненту User.vue та UserResults.vue, і описує загальну інформацію про користувача. Вкладений компонент UserResults.vue описує досягнення користувача в системі. Описує кількість виконаних вправ з кожного кодеку, а також список усіх пройдених тестувань.

Компонент CodePage.vue описує сторінку вибраного кодеку. Він містить загальну інформацію про вибраний код, а також містить три абстрактних динамічних компоненти: CodePractice.vue, CodeTheory.vue, CodeTest.vue. При виборі конкретного коду, замість абстрактних компонентів підставляються конкретні. Навігація по компонентам здійснюється за допомогою табуляції через компонент Vuetify v-tabs. Вкладені компоненти підтримуються активними та не знищуються за допомогою директиви Vue keep-alive. Таким чином, користувач може вільно переключатися між теоретичними відомостями та практичним завданням.

Практичний розділ додатку зображено на рисунку 8.2.

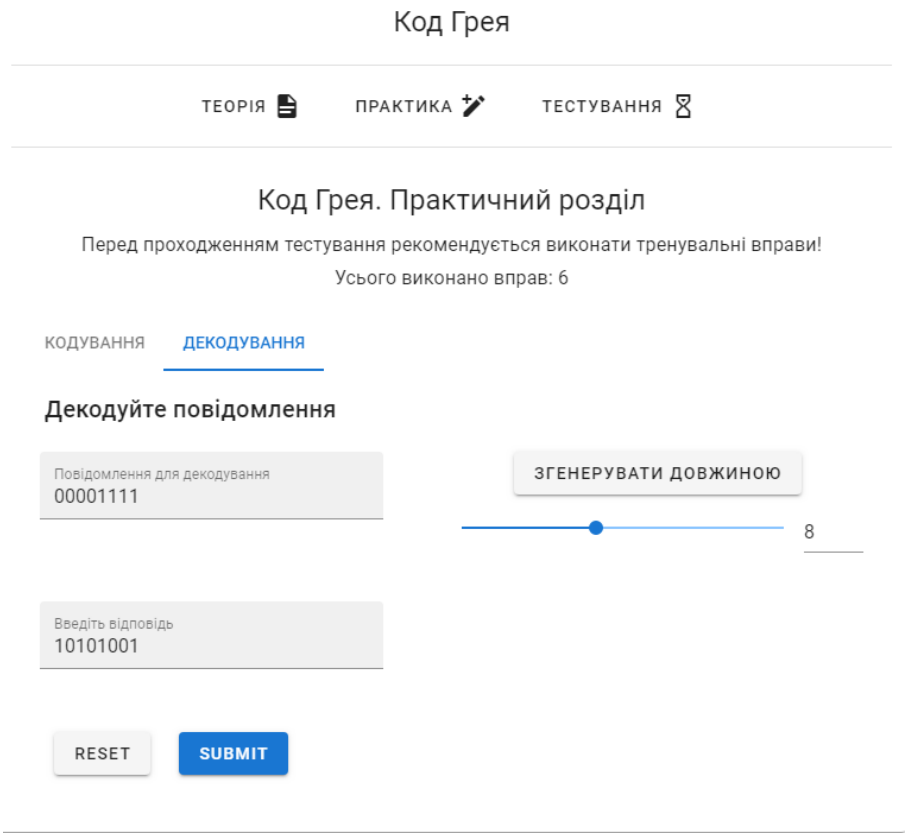

Рисунок 8.2 – Практичний розділ додатку

Компонент InfoPage.vue описує сторінку довідки та містить додаткову інформацію для користувача. Вона надає поради та інструкції для користування системою, дані про розробку системи.

8.2 Маршрутизація додатку з використанням VueRouter

Для створення виду багатосторінкового додатку, при розробці даної системи було використано маршрутизатор VueRouter. Ця частина екосистеми Vue дозволяє поставити у відповідність кожному URL-шляху відповідну веб-сторінку. При зміні пошукової адреси, VueRouter автоматично виконає дії для створення необхідного представлення [29].

Конфігурація маршрутизатора реалізована в файлу router.js і належить пакету

router. Під час налаштування маршрутизатора, відбувається вибір типу обробки історії та підключення до кореневого екземпляру Vue. В окремому файлі були описані і налаштовані використані шляхи додатку. При налаштуванні шляху використовується об'єкт Router, який в якості параметра приймає анонімний об'єкт, який містить масив routes. Цей масив містить перелік об'єктів, які описують конкретний маршрут застосунку. Кожний об'єкт має обов'язкові властивості: path, яка зберігає URL-адресу, name, яке можна використати для швидкого звернення до шляху та компонент який буде відображено при переході на відповідний шлях. Зазвичай, для конфігурації маршрутів використовуються компоненти, які описують повну веб-сторінку.

Для динамічного відображення за допомогою VueRouter, використовується компонент router-view, в який автоматично вбудовується компонент з активного в даний момент шляху.

Оскільки розроблювана система не є об'ємною, але розгалуженою, шляхів було використано небагато. Для обробки процесу авторизації використано наступні шляхи:

- auth/register;
- auth/login;
- auth/logout;

Ці маршрути описують відповідно сторінки реєстрації, авторизації, виходу з облікового запису. Для виходу з облікового запису звісно не існує окремої виділеної сторінки, проте даний маршрут використано для обробки події. Також, при виході з запису користувачем, для даного маршруту існують навігаційні захисники, функції які виконуються перед тим як відбувається перехід на маршрут.

Навігаційні захисники слугують для перевірки стану додатку і визначають, чи можливий перехід на бажаний шлях. В даному випадку, вони перевіряють, чи дійсно користувач авторизований і збережений в менеджері стану. При проходженні функції, відбувається виклик методі пакету auth, які здійснюють логіку авторизації. А при відміні користувач буде повернений на попередній шлях, якщо це можливо, або буде висвітлено повідомлення про необхідність авторизації для доступу. Такі

навігаційні захисники використовуються для всіх маршрутів, які пов'язані з функціоналом системи, наприклад виконання тестування або перегляду своїх оцінок. Лише авторизований користувач повинен мати доступ до певних функцій системи. Таким чином, навігаційні захисники дозволяють зручно обмежити доступ до вразливих частин додатку.

Для зображення основного функціоналу системи були реалізовані наступні маршрути:

- /home;
- /info;
- /user/:id;
- /user/:id/results;
- /codes;
- /codes/:name.

Маршрути /home і /info відображають головну сторінку та сторінку з інформацією про додаток відповідно. Вони використовують компоненти HomePage.vue та InfoPage.vue, для відображення контенту. Як і всі нижче перераховані маршрути, вони використовують навігаційні захисники для захисту важливої інформації.

Шлях /user/:id використовує компонент UserPage.vue, і відображає сторінку користувача. Він містить загальну інформацію про обліковий запис користувача та дозволяє змінювати персональні дані і налаштовувати профіль. Також, цей шлях має дочірній маршрут /user/:id/results, з компонентом UserResults.vue. Використання дочірніх маршрутів і компонентів дозволяє покращити продуктивність додатку, оскільки буде змінена не вся веб-сторінка, а лише її частина. В цьому компонентів користувач зможе ознайомитися з своїми результатами, отриманими за пройдені тестування. Сторінка результатів користувача зображена на рисунку 8.3.

Шлях /codes містить відповідний компонент CodeList.vue, і описує список усіх доступних для роботи кодів, які представлені у вигляді router-link, спеціального компоненту, який надає VueRouter. Router-link дозволяє замаскувати посилання для переходу на інший шлях під будь-який елемент користувацького інтерфейсу.

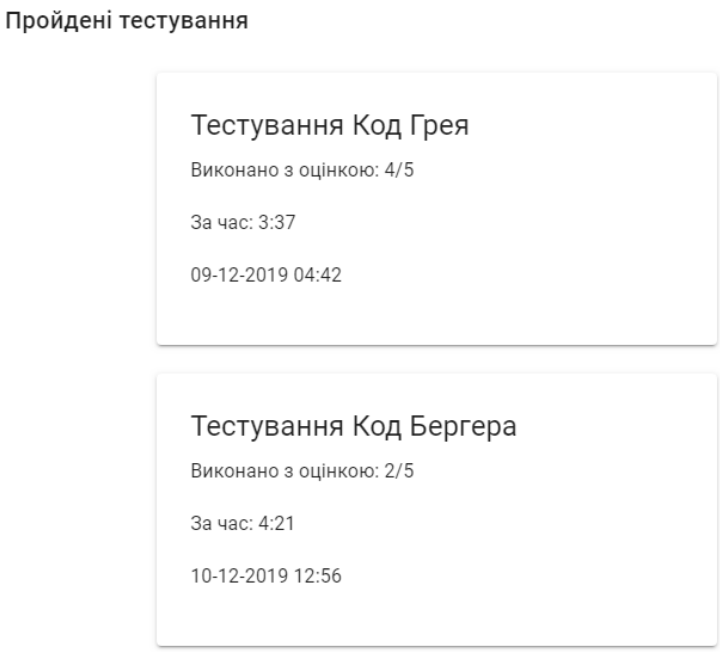

Рисунок 8.3 – Сторінка результатів користувача

Завдяки широким можливостям налаштування, для даної системи router-link було використано в якості основного посилання. За його допомогою, при виборі конкретного коду, наприклад коду Бергера зі списку, router-link автоматично вибере назву зі списку наявних в системі кодів, перевірить, чи є такий шлях, а потім перенаправить користувача на шлях /codes/BergerCode, де йому буде представлена сторінка і функціонал коду Бергера.

Перевагою VueRouter є легкість конфігурації динамічних маршрутів. Для правильного налаштування і відображення десятків сторінок з різними кодами використовується один динамічний шлях /codes/:name, який відображає абстрактний компонент Code.vue Цей компонент складається з трьох дочірніх компонентів, які відображають функціонал кодеків: теоретичні відомості про даний код, практичні завдання та тестування, або CodeTheory.vue, CodePractice.vue, CodeTest.vue відповідно. Приклад компоненту тестування зображено на рисунку 8.4.

Ці компоненти містять лише базовий користувацький інтерфейс, і замінюються конкретними при переході на маршрут. Після переходу на вибраний динамічний шлях, наприклад /codes/BergerCode, VueRouter відобразить компонент Code.

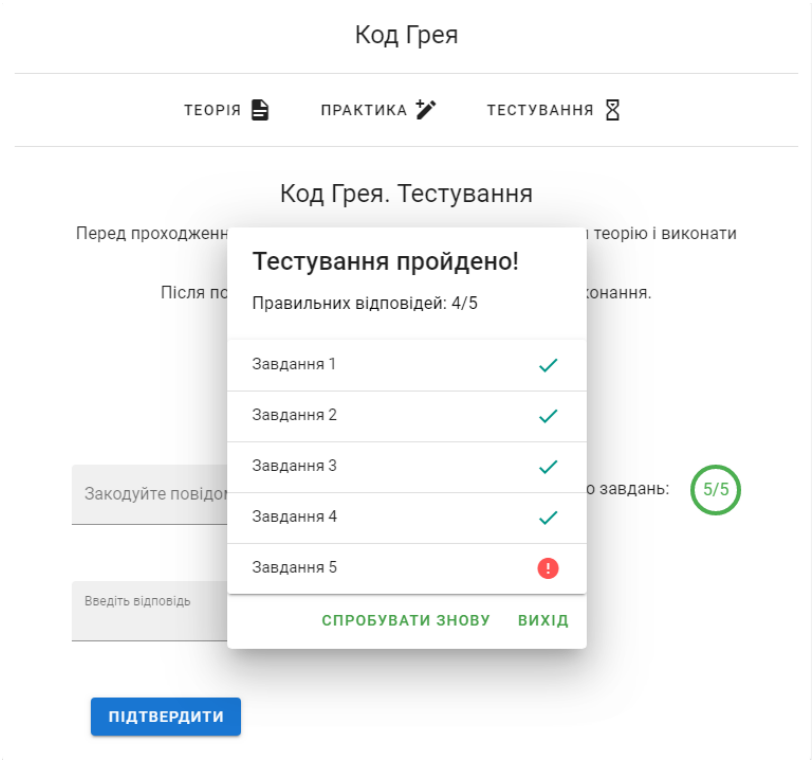

Рисунок 8.4 – Компонент тестування для коду Грея

При його створенні, Vue здійснить пошук по необхідним компонентам, та скомпілює BergerTheory.vue, BergerPractice.vue, BergerTest.vue. Більш детально систему маршрутизації шляхів зображено у додатку И.

8.3 Висновки до розділу

В даному розділі було розглянуто реалізацію користувацького інтерфейсу системи. Для готових компонентів було використано бібліотеку Vuetify, яка реалізована з використанням Material Design. Vuetify була встановлена до системи в якості плагіну. Додаткові залежності, такі як шрифт Google Roboto та іконки Material Design Icons, були встановлені, використовуючи CDN. Для побудови потрібних сторінок були використані компоненти розмітки, групування, форми та кнопки. На їх основі були побудовані власні компоненти.

Реалізовані компоненти представлення поділені на пакети views та components. Views описує компоненти, які представляють собою веб-сторінки додатку. До пакету

components належать компоненти, які описують окремі елементи користувацького інтерфейсу, окремі форми, панель навігації, меню та інші. З них складаються веб-сторінки.

Для маршрутизації односторінкового додатку був використаний VueRouter. З його допомогою створюється функціонал переходу між елементами застосунку. У відповідність URL-шляху ставиться компонент Vue. При переході на цю адресу, VueRouter автоматично скомпілює і відобразить потрібний компонент. Для розгалуженої системи з багатьма вкладеними компонентами, було створено дочірні шляхи, які забезпечують можливість переходу на різні частини додатку, без перевантаження основної частини.

9 РОЗРОБЛЕННЯ СТАРТАП-ПРОЕКТУ

Стартапом є унікальна ідея, орієнтовану на комерційних успіх. Вона не копіює вже існуючі проекти, а пропонує клієнтам дещо нове. Зазвичай, стартап-проект повинен привносити в сферу якусь інновацію, що аргументує доцільність вкладених в нього ресурсів, як майнових і фізичних так і часу. Проте, в більшості випадків, запропонувати потенційним користувачем щось зовсім новаторське доволі складно, і потребує великих вкладень. Тому Інтернет отримав широку популярність як стартова площадка для запуску, розвитку і функціонування стартап-проекту. Для таких стартапів важлива нова ідея, подача, обслуговування, підхід, який дозволить вирішити існуючі проблеми.

9.1 Опис ідеї проекту

Основна ідея стартап-проекту описана в таблиці 9.1.

Таблиця 9.1 – Опис ідеї стартап-проекту

Зміст ідеї	Напрямки використання	Вигоди для користувача
	1. Виконання практичних завдань студентами шкіл на ВНЗ	Покращення навчального процесу виконання і перевірки практичних завдань
	2. Вивчення алгоритмів роботи кодеків на процесі моделювання	Заміна застарілих технологічних рішень на більш сучасні

Основними техніко-економічними характеристиками ідеї є:

- кросплатформеність додатку;
- доступ до функціоналу системи онлайн;
- можливість збереження інформації користувача в системі;

- багаті можливості для налаштування процесу моделювання;
- можливість розширення системи новими модулями;
- доступ до теоретичних відомостей;
- можливість використання функціоналу системи в навчальних цілях;
- наявність українського перекладу.

В якості конкурентів можна визначити дві схожі програми: система моделювання кодеків XTest+, та система для моделювання та обробки цифрових сигналів SignalJ. Для формування конкурентоспроможності розроблюваної системи, були визначені її слабкі, сильні та нейтральні характеристики шляхом визначення конкурентів та аналізу інформації щодо їх техніко-економічних характеристик. Порівняння розроблюваної системи з конкурентами зображено на таблиці 9.2.

Таблиця 9.2 – Визначення сильних, слабких та нейтральних характеристик ідеї проекту.

№	Техніко-економічні характеристики ідеї	Потенційні товари/концепції конкурентів			W (Слабка сторона)	N (Нейтральна сторона)	S (Сильна сторона)
		Мій проект	XTest+	SignalJ			
1.	Кросплатформеність додатку	Має	Має	Має	-	-	+
2.	Доступ до функціоналу системи онлайн	Має	Немає	Немає	+	-	-
3.	Можливість збереження системи інформації користувача в системі	Має	Має	Має	-	-	+

Продовження таблиці 9.2

№	Техніко-економічні характеристики ідеї	Потенційні товари/концепції конкурентів			W (Слабка сторона)	N (Нейтральна сторона)	S (Сильна сторона)
		Мій проект	XTest +	SignalJ			
4.	Багаті можливості налаштування процесу моделювання	Має	Немає	Має	-	-	+
5.	Можливість розширення системи	Має	Має	Немає	-	+	-
6.	Доступ до теоретичних відомостей	Має	Має	Немає	-	-	+
7.	Можливість використання функціоналу системи в навчальних цілях	Має	Має	Немає	-	+	-
8.	Наявність українського перекладу	Має	Немає	Має	-	+	-

9.2 Технологічний аудит ідеї проекту

Для розробки проекту, необхідно провести аналіз технологій, які будуть використані для реалізації. Був обраний необхідний стек технологій, встановлена їх доступність. Таким чином була встановлена технологічна здійсненність ідеї проекту, яка зображена на таблиці 9.3

Таблиця 9.3 – Технологічна здійсненність ідеї проекту.

№	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
1.	Авторизація користувача	Сервіс OAuth2	Наявна	Доступна
2.	Збереження даних користувача	База даних Cloud Firestore	Наявна	Доступна
3.	Реалізація бізнес-логіки та користувацького інтерфейсу	Веб-фреймворк Vue.js, мова програмування JavaScript	Наявна	Доступна
4.	Серверна частина та зв'язок до неї	Firebase, проткол HTTP, vue-resource	Наявна	Доступна
Обрана технологія реалізації ідеї проекту: є можливою.				

Згідно з проведеним дослідженням, вибрані технології знаходяться у вільному доступі, легко доступні до використання, методи реалізації є доступними.

9.3 Аналіз ринкових можливостей запуску стартап-проекту

Для подальшого запуску стартап-проекту у ринкове середовище, необхідно визначити ринкові загрози, спланувати напрями розвитку, визначити стан ринку, потреби потенційних клієнтів та пропозицій конкурентів. Характеристика потенційного ринку зображена на таблиці 9.4.

Таблиця 9.4 – Попередня характеристика потенційного ринку стартап-проекту

№	Показники ринку (найменування)	Характеристика
1.	Кількість головних гравців, од	4 (розробник, ВНЗ, школи, студенти)
2.	Загальний обсяг продаж, грн/ум. од.	Понад 4000
3.	Динаміка ринку (якісна ціна)	Зростає

Продовження таблиці 9.4

4.	Наявність обмежень для входу	Отримання відповідної ліцензії від Міністерства Освіти України для шкіл, узгодження за навчальною програмою ВНЗ
5.	Специфічні вимоги до стандартизації та сертифікації	Відповідно до стандартів впровадження навчального процесу у школах і ВНЗ
6.	Середня норма рентабельності в галузі (або по ринку), %	125%

За результатами проведеного дослідження, ринок є привабливим для входження, а норма рентабельності є задовільною. Далі, необхідно визначити потенційних клієнтів, їх характеристики та сформувати перелік вимог до кожного товару. Характеристика описана в таблиці 9.5.

Таблиця 9.5 – Характеристика потенційних клієнтів стартап-проекту

Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
Потреба в якісному і сучасному продукті, яке дозволяє зробити навчальний процес швидким і якісним	Вищі навчальні заклади, школи, студенти	Використані при розробці технології, новий функціонал порівняний зі старим.	висока якість продукту, легкість у використанні, відповідність навчальним стандартам, низька ціна

Після проведення аналізу стосовно потенційних груп клієнтів, було встановлено, що основна частина клієнтів знаходиться серед студентів вищих та середніх навчальних закладів. Тому важливо зосередитись на їх основних вимогах, в першу чергу низька ціна, оскільки більшість студентів не мають великих коштів, які можуть витрачати на такі товари. Після визначення вимог клієнтів необхідно провести огляд ринку, розглянути фактори, які сприяють ринковому впровадженню проекту, та факторів, що йому перешкоджають (табл. 9.6, табл. 9.7).

Таблиця 9.6 – Фактори загроз стартап-проекту

№	Фактор	Зміст загрози	Можлива реакція компанії
1.	Застарілість поглядів щодо навчального процесу	Неможливість введення нового програмного забезпечення для кінцевих користувачів	Зменшення цін, пошук нових клієнтів
2.	Виявлення нових технологій в наслідку наукового прориву	Поява нових кодів, покращення або зміна старих.	Додання нових технологій до продукту
3.	Поява нової системи-конкурента на ринку	Поява конкурентів	Аналіз продукту конкурента, адаптація проекту
4.	Зміна програми курсу для ВНЗ	Програмний продукт виявляється непотрібним	Адаптація до нової програми
5.	Зміна ліцензій на використані при розробці технології	Неможливість використання технологій, які були використані при розробці продукту	Зміна використаних технологій на аналогічні
6.	Перехід на нові техніки навчання	Нове програмне забезпечення стає застарілим і непотрібним	Впровадження нових підходів та технік в продукт

Таблиця 9.7 – Фактори можливостей стартап-проекту

№	Фактор	Зміст можливості	Можлива реакція компанії
1.	Зацікавленість в продукті іноземних навчальних закладів	Розширення ринку, нові клієнти	Дороблення програмного продукту для вимог нового клієнта
2.	Зацікавленість у продукті спонсорів або інвесторів	Збільшення капіталу проекту	Масштабування програмного продукту, поширена реклама
3.	Включення програмного продукту до загального електронного навчального комплексу	Вигідна інтеграція в національний продукт	Переробка частини функціоналу для інтеграції в якості модуля в велику систему.
4.	Запровадження програмного продукту у закладах середньої освіти	Розширення ринку, залучення нових клієнтів	Адаптація програмного продукту до шкільного навчального рівня
5.	Додання до системи нових програмних модулів	Розширення системи призведе до збільшення зацікавлених клієнтів	Розширення програмного продукту

Для проекту були встановлені основні фактори загроз і можливостей. Більшість загроз пов'язані з появою нових або розвитком старих технологій, що сприяє появі нових конкурентів і змушує проект адаптуватися. Проте, було встановлено, що загрози не є критичними, а шанси їх виникнення є доволі низькими. Можливості, які сприяють ринковому впровадженні, призводять до зростання попиту і відповідно масштабування проекту. Позитивні можливості загалом переважують ризики загроз. Таким чином, проект має гідний потенціал розвитку в ринковому середовищі. Тому, наступним кроком було проаналізовано існуючу конкуренцію на ринку (табл. 9.8).

Таблиця 9.8 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив діяльності підприємства
Тип конкуренції: олігополія	Конкурентами є кілька застарілих програмних продуктів	Залучення зацікавлених лиць для покращення загальної якості і конкурентоспроможності
За рівнем конкурентної боротьби: національний	Ринок охоплює вищі і середні навчальні заклади	Забезпечення відповідності програмному продукту навчальним стандартам
За галузевою ознакою: внутрішньо-галузева	Конкуренти знаходяться в одній галузі (програмне забезпечення)	Аналіз факторів, які впливають на успіх у даній галузі
Конкуренція за видами товарів: товарно-видова	Товар конкурентів одного виду – програмне забезпечення	Проект орієнтований на малий і середній бізнес, тому необхідний вихід на ринок в даних областях
За характером конкурентних переваг: не цінова	Товар якісніший за продукти конкурентів	Зосередження ресурсів на підтриманні вищої якості продукту
За інтенсивністю: не марочна	Товар тільки виходить на ринок	Забезпечити успішний старт продукту на ринку

Після ступеневого аналізу конкуренції на ринку, стає очевидним, що конкурентні товари, хоч і утримують стійку олігополію, мають значні недоліки, використовуючи які, проект може значно посилити свої позиції на ринку. Оскільки проект орієнтований на малий та середній бізнес, ризики та рівень входження є незначними. Наступним кроком було здійснено більш детальний аналіз умов конкуренції в галузі (табл. 9.9).

Таблиця 9.9 – Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потецнійні конкуренти	Постачальники	Клієнти	Товари-замінники
	Застарілі системи моделювання, XTest+, SignalJ	Конкурентом може стати будь-який продукт в сфері моделювання	Арендатори хостингових, хмарних сервісів, які використовуються в продукті	Студенти вищих та середніх навчальних закладів	Не беручи до уваги конкурентів, товарів-замінників на даний момент не існує
Висновки	Конкуренція існує, проте її інтенсивність доволі низка	Потенційних конкурентів доволі мало, проте можуть виникнути під час впровадження продукту	Існує невелика залежність від постачальників, проте вони не диктують умови роботи продукту.	Головна умова клієнтів відповідність до навчальних стандартів і легкість у використанні	Таким чином, обмежень на ринку через товари-замінники не існує

Аналіз за М. Портером показав, що на даний момент в часі існують лише прямі конкуренти, виникнення потенційних є малоймовірним, товарів-замінників не існує. Тобто, конкурентна ситуація на ринку сприятлива і проект може мати успіх. Проте, для цього необхідно встановити ключові сильні сторони, які повинен мати проект, щоби бути конкурентоспроможним на ринку. Обґрунтування факторів конкурентоспроможності описано в таблиці 9.10.

Таблиця 9.10 – Обґрунтування факторів конкурентоспроможності

№	Фактор конкурентоспроможності	Обґрунтування факторів
1.	Зручність використання	Продукт зменшує час на виконання лабораторної роботи та покращує навчальний процес
2.	Комплексний підхід	Поєднання усього функціоналу в одному місці покращує взаємодію користувача з системою
3.	Можливість розширення продукту	Продукт передбачає можливість розширення додатку та завбачає гнучкість навчальної програми
4.	Легкість інтеграції в навчальний процес	Інтеграція продукту передбачає надання студентам доступу до веб-застосунку, що легко виконати і відмінити

Для досягнення успіху на ринку, для проекту були обрані фактори, на яких буде заснована його перевага над конкурентами. Найважливішими серед них можна виділити легкість інтеграції і можливість розширення. Оскільки основними конкурентами є застаріле програмне забезпечення, новий продукт повинен з замінити старий. Наступним кроком був проведений аналіз сильних та слабких сторін стартап-проекту (табл. 9.11).

Таблиця 9.11 – Порівняльний аналіз сильних і слабких сторін стартап-проекту

Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів порівняно з проектом						
		-3	-2	-1	0	+1	+2	+3
Зручність використання	6			XTest+			SignalJ	
Комплексний підхід	10			SignalJ		XTest+		
Можливість розширення	5		SignalJ			XTest+		
Легкість інтеграції в навчальний процес	12			XTest+		SignalJ		

Порівняльний аналіз продемонстрував, що за факторами конкурентоспроможності, стартап-проект має значні переваги над конкурентами і тому може мати успіх на ринку. Для повноти аналізу, необхідно складання SWOT-аналізу на основі ринкових можливостей та загроз, а також сильних і слабких сторін (табл. 9.12).

Таблиця 9.12 – SWOT-аналіз стартап-проекту

Сильні сторони: - новизна проекту; - легкість інтеграції в навчальний процес; - відповідність продукту навчальним стандартам; - наявність потреби у сучасних продуктах в галузі; - витіснення неякісних, застарілих продуктів та сприяння конкуренції.	Слабкі сторони: - несприятливе ринкове середовище; - складність масштабного впровадження; - відносна залежність від постачальників.
Можливості: - розширення ринку в наслідок зацікавленості в продукті; - поява інвесторів; - послаблення позицій конкурентів; - поява новітніх технологій.	Загрози: - застарілість поглядів щодо використання нових продуктів; - захоплення ринку новими сильними конкурентами; - зміна курсу навчальних закладів; - зміна ліцензій і договорів постачальників.

SWOT-аналіз дав змогу комплексно оцінити і порівняти ключові сторони проекту та ринкового середовища. Було встановлено, що сильні сторони, особливо новизна та легкість інтеграції, легко переважають слабкі сторони проекту. При поступовому впровадженні на ринку та розвитку проекту, недоліки будуть легко

виправлені і усунені. Це підкріплюється значними ринковими можливостями проекту, які ще більше сприяють його успіху серед конкурентів. Збитки через загрози, шанси появи яких є незначними, можна усунути невеликими витратами. Після проведення SWOT-аналізу, були розроблені альтернативи ринкової поведінки для виведення стартап-проекту на ринок (табл. 9.13).

Таблиця 9.13 – Альтернативи ринкового впровадження

Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
Загарбник	Значна	Максимум рік
Наступник	Суттєва	Максимум рік

Для даного стартап-проекту були обрані дві альтернативи ринкової поведінки – загарбник і наступник. Наступник підходить оскільки проект орієнтований на невеликий сегмент ринку, на самому ринку конкуренція представлена у вигляді олігополії. Використання даної альтернативи дозволить спеціалізуватися на одній частині ринку, а потім поглиблювати свій вплив. Іншою альтернативою є загарбник, який орієнтований на захоплення ринку від конкурентів.

9.4 Розроблення ринкової стратегії проекту

Для забезпечення досягнення поставленої мети стартап-проекту у ринковому середовищі, необхідно розробити ринкову стратегію, сукупність заходів, які спрямовані на отримання запланованих обсягів продажу і прибутку. Перший крок полягає у визначенні стратегії охоплення ринку. Для цього був проведений аналіз цільових груп потенційних споживачів, їх очікування від продукту, конкуренції в різних групах та наскільки прости вхід в ринковий сегмент. (табл. 9.14).

Таблиця 9.14 – Вибір цільових груп потенційних споживачів

Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтований попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу в сегмент
Вищі навчальні заклади	Висока	Високий	Висока	Середня
Середні навчальні заклади	Середня	Середній	Низька	Середня
Які цільові групи обрано: вищі і середні навчальні заклади				

Серед потенційних споживачів були розглянуті вищі та середні навчальні заклади. Оскільки фінансування та технічне обладнання вищих навчальних закладів вище ніж у школах, основний пріоритет відданий університетам. Проте повністю відмовлятися від середніх навчальних закладах не є раціональним.

За результатом аналізу потенційних груп споживачів, була визначена стратегія охоплення ринку. Оскільки потенційні споживачі належать одному сегменту ринку, доцільно вибрати стратегію концентрованого маркетингу. Наступний крок розробки ринкової стратегії полягає в визначенні базової стратегії розвитку (табл. 9.15).

Таблиця 9.15 – Визначення базової стратегії розвитку

Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
Наступник	Концентрація на одному цільовому сегменту ринку	Надання клієнтам сучасного продукту для навчальних цілей	Стратегія спеціалізації

З наявних базових стратегій розвитку була обрана стратегія спеціалізації, як найбільш підходяща. Вона передбачає концентрацію на одному сегменту ринку, і задоволенню потреб цього сегмента краще ніж конкуренти. Оскільки стартап-проект зосереджена на перевазі над застарілими конкурентами, така стратегія є підходящою. Наступним пунктом в розробці ринкової стратегії було вибрано стратегію конкурентної поведінки (табл. 9.16).

Таблиця 9.16 – Визначення базової стратегії конкурентної поведінки

Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
Проект є наступником вже існуючого застарілого товару	В першу чергу проект витіснить існуючих конкурентів, потім буде шукати нових споживачів	Копіювання таких характеристик як: можливість розширення, легкість інтеграції, кросплатформеність продукту	Стратегія виклику лідера.

З наявних стратегій конкурентної поведінки була обрана стратегія виклику лідера. Вона полягає в активній конкуренції з лідерами ринку з метою зайняти їх місце. Це виправдано конкурентним середовищем на ринку, де панує олігополія, а також перевагами проекту над конкурентами. Тому така дещо ризикована стратегія є обґрунтованою. Наступним кроком було визначено стратегію позиціонування проекту (табл. 9.17).

Таблиця 9.17 – Визначення стратегії позиціонування

Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
Зручний інтерфейс, збереження даних, кросплатформеність, можливість доступу до системи онлайн	Стратегія спеціалізації	Стратегія виклику лідера	Можливість розширення системи, відповідність навчальним стандартам, легкість інтеграції

Для визначення стратегії позиціонування було використано основні сильні сторони стартап-проекту: зручний інтерфейс, можливість збереження даних, кросплатформеність додатку. Ці фактори високо оцінені потенційними клієнтами, а також відсутні у конкурентів. Тому, завдяки ним, була досягнута перевага над продуктами конкурентів. Сильні сторони проекту також були використані в якості факторів конкурентноспроможності стартап-проекту.

9.5 Розроблення маркетингової програми стартап-проекту

Для успішної реалізації стартап-проекту було розроблено маркетингову програму, тобто комплекс завдань виробничого та організаційного характеру, з визначенням ресурсів, що використовуються. Першим кроком є формування маркетингової концепції товару, який отримає споживач.

Для цього були визначені основні очікувані потреби потенційних користувачів, та відповідні вигоди, які пропонує стартап-проект. Ці вигоди також є основними перевагами на товарами конкурентів. Таким чином, було визначені ключові переваги концепції потенційного товару (табл. 9.18).

Таблиця 9.18 – Визначення ключових переваг концепції потенційного товару

Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
Потреба в якісному програмному забезпеченні для моделювання кодеків	Сучасний, зручний у використані програмний продукт, який замінює існуючі застарілі аналоги та надає багатий вибір нових функцій	Кросплатформеність додатку, легкість використання, можливість збереження даних користувача
Потреба в введення нових методик в навчальному процесі	Легкість інтеграції продукту в навчальний процес, відповідність навчальним стандартам	Наявність українського перекладу, можливість виконання практичних завдань, наявність теорії та додаткового матеріалу, можливість проходження тестів, і їх адаптації до стандартів навчальних закладів

Основними перевагами концепції потенційного товару є його багатофункціональність та зручність у використані, порівняно з конкурентами. Також великою перевагою є легкість інтеграції програмного продукту у навчальний процес при заміні технологій. Це обумовлюється спеціалізацією стартап-проекту на навчальну складову і відповідність освітнім стандартам.

Наступним кроком розробки маркетингової програми було побудовано трирівневу маркетингову модель товару, метою якої є загальний опис послуг, які надаються товаром. Модель зображено на таблиці 9.19.

Таблиця 9.19 – Опис трьох рівнів моделі товару

<table>
<tr><td>Рівні товару</td><td colspan="3">Сутність та складові</td></tr>
<tr><td></td><td colspan="3">Опис базової потреби споживача, яку задовольняє товар (згідно концепції), її основної функціональної вигоди</td></tr>
<tr><td></td><td colspan="3">Надання сучасного програмного забезпечення для моделювання кодеків, що дозволить значно покращити навчальний процес</td></tr>
<tr><td rowspan="6"></td><td>Властивості/ характеристики</td><td>М/Нм</td><td>Вр/Тх/Тл/Е/Ор</td></tr>
<tr><td>1. Економічні: зменшення ціни на обслуговування та експлуатацію
2. Технологічні: використання сучасних технологій
3. Ергономічність: зручний користувацький інтерфейс, зрозумілий функціонал продукту
4. Естетичні: привабливий та якісний дизайн продукту
5. Безпеки: відповідність нормативам використання електронних пристроїв та програмного забезпечення</td><td>–/+</td><td>+/+/+/+/+</td></tr>
<tr><td colspan="3">Якість: стандарти та нормативи Міністерства Освіти України для навчальних закладів</td></tr>
<tr><td colspan="3">Документи виконані з логотипом підприємства</td></tr>
<tr><td colspan="3">Марка: TikTest</td></tr>
<tr><td></td><td colspan="3">До продажу: представлення клієнтам продукту</td></tr>
<tr><td></td><td colspan="3">Після продажу: Допомога в налаштуванні і підтримка продукту</td></tr>
<tr><td colspan="4">За рахунок чого потенційний товар буде захищено від копіювання: захищення інтелектуальної власності шляхом патентування</td></tr>
</table>

Після формування маркетингової моделі товару наступним кроком є визначення цінових меж, якими необхідно керуватись при встановлені ціни на потенційний товар (табл. 9.20).

Таблиця 9.20 – Визначення меж встановлення ціни

Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
100-200 ум. од.	100-200 ум. од.	700-800 ум. од.	80-180 ум. од.

В орієнтовну ціну також було включено ціна послуги постачальників. Таким чином, потенційний товар є доступним для цільової групи споживачів, проте є дешевшим за конкурентів. Наступним кроком є формування системи збуту (9.21).

Таблиця 9.21 – Формування системи збуту

Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
Замовлення проекту	Швидкість, виконання, надійність (надання консультацій та підтримка проекту)	Глибока	Власні сили, зацікавлені навчальні заклади

Для оптимальної системи збуту було вирішено проводити збут власними силами, розповсюджуючи його у навчальні заклади на умові замовлень. Проте, навчальні заклади можуть також допомогти в розповсюдженні товару. Останньою складової маркетингової програми є розроблення маркетингових комунікацій (табл. 9.22).

Таблиця 9.22 – Концепція маркетингових комунікацій

Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламних повідомлень	Концепція рекламного звернення
Цільові клієнти проводять багато часу у соціальних мережах та Інтернеті, проте також відвідують тематичні конференції та технічні співтовариства	Інтернет, соціальні мережі, технічні співтовариства, виставки і конференції	В мережі Інтернет і відповідних соціальних мережах буде здійснюватися рекламна розсилка, у технічних співтовариствах буде розповсюджуватися друкована продукція	Донести до користувачів інформацію про існуючий більш якісний і сучасний аналог застарілого продукту	Рекламне звернення в доброзичливій манері сповіщає користувачів про появу нового продукту на ринку

Основними каналами комунікаціями, якими користуються цільові клієнти, є Інтернет і соціальні мережі, тому використання таких каналів є виправданим. Також, маркетинг в Інтернеті дозволяє охопити більш широку аудиторію і зацікавити більше клієнтів.

9.6 Висновки до розділу

В даному розділі був розглянутий потенціал висунення розроблюваної системи в якості стартап-проекту, була обґрунтована її конкурентоспроможність, проведений

аналіз можливих виходів на ринок, різних стратегій розвитку та маркетингу.

Були проаналізовані основні техніко-економічні характеристики ідеї, чим вона відрізнялась від існуючих аналогів. З них були виділені: доступ до системи онлайн, можливість розширення системи новими модулями та кросплатформеність додатку. Порівнюючи ці характеристики з аналогами, було встановлено, що ідея є конкурентоспроможною, і може мати потенційний успіх на ринку.

Також був проведений аудит технологій, за допомогою яких можна реалізувати ідею проекту. З основних були виділені: мова програмування JavaScript, веб-фреймворк Vue.js, та Firebase. Використання цих технологій є в цілому безкоштовним та доступним для розробки проекту.

Був проведений аналіз ринкових можливостей запуску стартап-проекту. Для цього була розглянута характеристика потенційного ринку, стан конкуренції, ринкове впровадження стартап-проекту. Згідно з проведеним аналізом, була встановлена, що проект має значні переваги над аналогами, а отже є конкурентоспроможним. Тому для проекту були обрані альтернативи ринкової поведінки.

Також були розроблені ринкові стратегії проекту. Для цього були проаналізовані групи потенційних клієнтів, і були обрані вищі і середні навчальні заклади. Враховуючи даний сегмент ринку, були вибрані стратегії розвитку спеціалізації та стратегія конкурентної поведінки виклику лідеру. Оскільки даний проект орієнтований на невеликий сегмент ринку, а також конкурентну олігополію, дані стратегії є виправданими.

Була розроблена маркетингова програма стартап-проекту, внаслідок чого було встановлені цінові межі на потенційний товар та сформовані системи збуту. Пріоритетом маркетингової програми було встановити низькі ціни для вибраного сегменту клієнтів, а також вибрати оптимальні канали збуту. Такими були вибрані Інтернет та соціальні мережі, оскільки ними користуються частіше за інші, а також ці канали дозволяють охопити більше аудиторії.

ВИСНОВКИ

В ході виконання магістерської роботи було розглянуто питання покращення шляхів дослідження роботи завадостійких кодеків. В результаті роботи була створена система для їх моделювання, яка покращує навчальні процеси вивчення, робить їх доступнішими та зручнішими для використання.

Для розроблення системи, було проаналізовано аналогічні продукти. В якості аналогів були вибрані навчальна система моделювання кодеків XTest+ та система моделювання та обробки сигналів SignalJ. Після аналізу було визначені основні проблеми, на виконання яких була зосереджена розроблювана система. Серед найбільших були виділені: збереження даних користувачів, забезпечення безпечного доступу до системи, мати інтерфейс для налаштування процесу моделювання.

Враховуючи проаналізовані недоліки аналогів, були визначені функціональні та нефункціональні вимоги до системи. Основні функціональні вимоги пов’язані з забезпеченні авторизації системи для збереженні даних користувача і його результатів роботи в додатку. Також, вимоги визначені для функціоналу системи: надання користувачу теоретичних відомостей, можливостей для виконання практичних завдань та проходження тестувань. Нефункціональні вимоги стосуються якості та продуктивності виконання системи, швидкості генерування завдань та безпомилкового оцінювання результатів.

Під час розробки системи, були визначені сценарії використання. Головними акторами були встановлені неавторизований та авторизовані користувачі. Неавторизований користувач не може отримати доступ до функціоналу системи, поки не здійснить авторизацію. Авторизований в системі користувач може вільно взаємодіяти з додатком, переглядами теоретичні відомості, виконувати практичні завдання, переглядати отримані результати та досягнення.

Для реалізації системи, було розглянуто існуючі технології та обрані найбільш підходящі. З врахуванням поставлених вимог, було вирішено реалізувати систему в якості безсерверного односторінкового застосунку з використанням Firebase. Таким чином, було забезпечено перенесення частини логіки авторизації та роботи з базою

даних на сторонній сервіс. Для реалізації клієнту був використаний веб-фреймворк Vue.js, оскільки він найбільше підходить для вирішення поставлених цілей. Також були використані основні частини екосистеми Vue: VueRouter, VueResource, Vuex, Vuetify. Основна розробка була виконана на мові програмування JavaScript.

Враховуючи вибрані технології, була розроблена структурна схема системи, яка складається з двох основних частин: сервісу Firebase, який виконує функцію серверу та односторінкового додатку Vue. Додаток має компоненту архітектуру, яку розбито на окремі модулі. З основних модулів можна виділити: модуль для взаємодії з Firebase, модуль, який містить логіку менеджеру стану та три модулі бізнес-логіки, для тестувань, користувача і функціоналу кодеків. Завершальний модуль містить компоненти представлення користувацького інтерфейсу. Окремо до системи підключені стороні пакети, так як бібліотека Vuetify і інші частини екосистеми Vue.

Для системи була використана нереляційна база даних Firebase, яка зберігає дані у форматі JSON і оновлює їх у реальному часі синхронізуючи клієнтів. Для встановлення сутностей, необхідних для реалізації системи, та зв'язків між ними була розроблена ER-діаграма бази даних. Основними сутностями були визначені: User, профіль користувача, CompletedTest, пройдений тест, ActiveTest, тест під час проходження, PracticeRecord, запис виконаних практичних завдань, TestTask, завдання для тестувань.

Основна частина бізнес-логіки була реалізована на клієнтській частині. Перенесення авторизації та бази даних на сервіс Firebase надало додаткову безпеку системі. Основний функціонал системи полягає в наданні користувачу можливості вивчати роботу кодеків, ознайомлюватися з теоретичними відомостями, виконувати практичні завдання та проходити тестування. Усі результати роботи користувача з системою зберігаються у базі даних. Бізнес логіка реалізована з використанням шаблону MVVM, який реалізований у кожному компоненті Vue.js. Також, були застосовані інші підходи розробки: поширення стану додатку через менеджер Vuex, виділення логіки в окремі файли і їх подальше підключення в багаторазових компонентах.

Інтерфейс користувача був розроблений з використанням бібліотеки Vuetify.

Бібліотека надає великий вибір готових компонентів для елементів користувацького інтерфейсу. З допомогою Vuetify були розроблені власні компоненти, які були розділені на пакети views, які містять цільні веб-сторінки та components, окремі елементи цих сторінок. Для маршрутизації застосунку використаний VueRouter. При налаштуванні шляхів, кожному URL-шляху відповідає компонент веб-сторінки. Таким чином, користувач отримує можливість переключатися між різними частинами додатку.

При розробленні системи був розглянутий потенціал висунення її в якості стартап-проекту. Були визначені слабкі та сильні сторони, проведений аналіз ринкових загроз та можливостей, визначені конкуренти та цільові групи клієнтів Було встановлено, що стартап-проект має можливості для закріплення на ринку та вибрана конкурента стратегія лідеру, на основі якої було розроблену маркетингову програму.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Полторак В.П. Теорія інформації та кодування / Ю.П. Жураковський, В.П. Полторак // Підручник. – К.: Вища школа, 2001. – 255 с.: іл., укр.мовою (Гриф МОНУ №1/11-2367 від 21.05.2001).
2. Полторак В.П. Калькулятор GF(q) для циклічних кодів / В.П. Полторак, Н.С. Вітіщенко // Вісник НТУУ \КПІ\. Інформатика, управління та обчислювальна техніка. Зб. наук. пр. Вип. 53. - К.: Век+, 2011. - С. 233 - 240.
3. ДСТУ 9241-11:2006 Ергономічні вимоги до роботи з відеотерміналами в офісі. Частина 11. Настанови щодо прийнятності у використанні.
4. Забезпечення якості. Функціональні та нефункціональні вимоги. [Електроний ресурс] – Доступ: http://lvivqaclub.blogspot.com/2008/10/blog-post_17.html
5. Нефункциональные требования к програмному обеспечению. Часть 1 [Електроний ресурс] – Доступ: https://habr.com/ru/post/231961/
6. Как и зачем использовать UseCases [Електроний ресурс] – Доступ: https://dou.ua/lenta/articles/use-cases/
7. Desktop Application vs Mobile App vs Web App [Електроний ресурс] – Доступ: https://www.miraclemobile.io/desktop-apps-vs-web-apps-vs-mobile-apps/
8. Desktop vs mobile: user experience optimization [Електроний ресурс] – Доступ: https://99designs.com/blog/web-digital/desktop-vs-mobile-app-design/
9. Web and Desktop Usage In 2019 [Електроний ресурс] – Доступ: https://www.perficientdigital.com/insights/our-research/mobile-and-desktop-usage-study
10. Web Application Architecture: How the Web Works [Електроний ресурс] – Доступ: https://www.altexsoft.com/blog/engineering/web-application-architecture-how-the-web-works/
11. How Single-Page Applications work [Електроний ресурс] – Доступ: https://blog.pshrmn.com/entry/how-single-page-applications-work/
12. jQuery Official API documentation [Електроний ресурс] – Доступ: https://api.jquery.com/

13. Почему я использую jQuery в 2019 [Електроний ресурс] – Доступ: https://webformyself.com/pochemu-ya-do-six-por-ispolzuyu-jquery-v-2019-godu/
14. Developer Survey Results 2018 [Електроний ресурс] – Доступ: https://insights.stackoverflow.com/survey/2018
15. Angular: Best Use Cases and Reasons To Opt For This Tool [Електроний ресурс] – Доступ: https://yalantis.com/blog/when-to-use-angular/
16. 7 Reasons why you should use React [Електроний ресурс] – Доступ: https://stories.jotform.com/7-reasons-why-you-should-use-react-ad420c634247
17. Introduction to Vue.js 2.0 [Електроний ресурс] – Доступ: https://vuejs.org/v2/guide/index.html
18. Vue.js with TypeScript [Електроний ресурс] – Доступ: https://johnpapa.net/vue-typescript/
19. Мова програмування JavaScript [Електроний ресурс] – Доступ: https://developer.mozilla.org/en-US/docs/Web/JavaScript
20. Uses of Firebase apps [Електроний ресурс] – Доступ: https://firebase.google.com/products/#develop-products
21. The Best Material Design Vue Framework [Електроний ресурс] – Доступ: https://medium.com/@johnmaeda/the-best-material-design-vue-js-framework-bd7e2b730b5a
22. Why Vuetify [Електроний ресурс] – Доступ: https://vuetifyjs.com/en/introduction/why-vuetify
23. Working with Webpack [Електроний ресурс] – Доступ: https://cli.vuejs.org/guide/webpack.html
24. Structure your data in Firebase [Електроний ресурс] – Доступ: https://www.airpair.com/firebase/posts/structuring-your-firebase-data
25. Structuring a Vue project Електроний ресурс] – Доступ: https://medium.com/@zitko/structuring-a-vue-project-87032e5bfe16
26. Гамма Э. Приемы объектно-ориентированного проектирования. Паттерны проектирования / Э. Гамма, Р. Хелм, Р. Джонсон. – К. : Питер, 2001. – 355 с.
27. Vue.js MVVM Concept Overview [Електроний ресурс] – Доступ:

https://012.vuejs.org/guide/mvvm

28. Vue architecture patterns [Електроний ресурс] – Доступ: https://medium.com/@ederng/the-vue-architecture-that-worked-for-me-in-small-and-large-apps-9b253cf92951
29. Vue Router Documentation [Електроний ресурс] – Доступ: https://router.vuejs.org/